(1)

A MICRO-COMPUTER BASED
DECISION SUPPORT SYSTEM
FOR
RESPONSE SURFACE METHODOLOGY

THESIS

David M. Leeper
Captain, USAF

Gregory J. Meidt
Captain, USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/ENS/GOR90M-12

A MICRO-COMPUTER BASED DECISION SUPPORT SYSTEM FOR RESPONSE SURFACE METHODOLOGY

THESIS

David M. Leeper
Captain, USAF

Gregory J. Meidt
Captain, USAF

ıber of annual Standard
ARE ONLY AVAILABLE

abstract

ESTIMATED
AVAILABILITY DATE

ST01. Prime Contractors with Awards Over $25,000 by Name, Location, and Contract Number — March 15

This tabulation is an alphabetical list of DoD prime contractors who received awards over $25,000, showing total dollar value for each contract being performed by a contractor at a place of performance. The tabulation is sorted in order of ultimate name, headquarters name, establishment name, place of performance, and contract number. If the prime contractor is a subsidiary of another firm, it will be listed under the name of the ultimate firm. (Approximately 4,200 pages)

ST06. — March 15

(Pr

This is a
FSC and
Developm
supplies
contracto
Departmeı
pages)

AFIT/ENS/GOR90M-12

A MICRO-COMPUTER BASED DECISION SUPPORT SYSTEM FOR RESPONSE SURFACE METHODOLOGY ANALYSIS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

David M. Leeper, B.S.
Captain, USAF

Gregory J. Meidt, B.S.
Captain, USAF

March 1990

Approved for public release; distribution unlimited

Acknowledgements

We could not have accomplished this task without the help and advice of many others. First, we are indebted to our advisors, Maj Bauer and Lt Col Valusek. Maj Bauer provided the expertise in RSM and focus to keep us on the right track. Lt Col Valusek provided the expert advise needed to create a decision support system. More importantly, we are indebted to God, our families, and friends who supported and guided us through the entire effort.

Table of Contents

List of Figures

List of Tables

AFIT/ENS/GOR90M-12

Abstract

Currently it can be difficult to conduct response surface methodology analysis because it requires computer assistance and a thorough understanding of the mathematics and statistical theory involved. The difficulty of RSM analysis would be greatly reduced if a system existed that integrated the RSM mathematics and the RSM decision making process into a single computer program.

The purpose of this ~~study~~ was to develop PCRSM, a personal computer based decision support system (DSS) for response surface methodology (RSM) analysis. This study focused on the experimental design and regression elements of the RSM analysis process.

Experimental design is conducted through a series of questions, choices, and advice regarding the numbers of factors, design choice, resolution, number of center points, and number of design replications. PCRSM includes 60 designs for 2 to 39 factors. It supports six different two and three level experimental designs, including full and fractional designs, Plackett-Burman designs, central composite designs, and Box-Behnken designs.

Least squares linear regression is used to determine the meta-model of the response surface. Output provided by the regression package includes residuals plots, two

different ANOVA tables, coefficient tables, variance-covariance matrices, correlation matrices, and the design matrix and responses.

In sum, PCRSM has transformed the currently cumbersome task of performing RSM analysis into an integrated computer program for the personal computer.

A MICRO-COMPUTER BASED DECISION SUPPORT SYSTEM FOR RESPONSE SURFACE METHODOLOGY ANALYSIS

I. INTRODUCTION

Response Surface Methodology

Response surface methodology (RSM) uses statistical techniques to describe the relationship between a system's inputs and outputs (called responses) (4:1). Many times a system, such as a computer simulation, may have many input and output parameters. Of these inputs, typically only a few have a significant impact on the outputs or responses. Response surface methodology attempts to isolate the significant inputs and characterize their relationship to the output. The end result is a relatively simple equation that can be used to describe, predict, or optimize system behavior.

Decision Support Systems

Currently it is not easy to conduct response surface methodology analysis because it requires computer assistance and a thorough understanding of the mathematics and statistical theory involved. The difficulty of RSM analysis would be greatly reduced if a system existed that integrated the RSM mathematics and the RSM decision making process into a single computer program. A decision support systems (DSS)

could accomplish this task. Decision support systems are computer based systems that help decision makers arrive at judgments and choices.

Objective

It is the purpose of this research to design and initiate development of PCRSM, a micro-computer based decision support system to support response surface methodology analysis.

Research Sub-objectives

There are several major sub-objectives involved in creating a decision support for response surface methodology analysis, including experimental design, regression, and optimization.

Decision Support Systems. As stated previously, decision support systems aid the decision maker's process of judgment and choice in making a decision. There are several decisions in the RSM analysis process, including selecting the input variables to investigate, identifying the proper output response, choosing an appropriate experimental design, and determining the correct model based on linear regression results. Each of these terms will be defined shortly.

Decision support systems have three components. They are the man-machine interface, models, and data base. The

man-machine interface is what the user sees when sitting at the computer, including menus, screen displays, and graphics. The computer models interact with the data base and user input to generate a representation of a particular system. In this research, the primary models include an experimental design expert system and a regression model. The data base manages the data stored and used by the DSS.

Sub-objectives.

1. Understand how decision support systems model a decision process.

2. Capture the RSM process and determine the decisions.

3. Select the part or kernel of the concept map to implement for the DSS.

4. Determine the user requirements and create the medium to store the requirements.

5. Create the correct models to support the decisions.

6. Create the correct data base to support the decisions.

7. Develop a man-machine interface (MMI) that is user friendly and correctly displays the needed information for the user. Additionally, develop the right screens and menus that allow the users to apply the DSS efficiently.

8. Provide the necessary interface to output the design matrix in the correct format for a regression package.

9. Provide the correct input interface to enter in the responses from the simulation model or other experimental output determined by the design matrix inputs.

Experimental Design. A goal of response surface methodology is to choose the inputs, or factors, that best describe the outputs. Dr. Jack Kleijnen describes experimental design as "a statistical technique for improving the efficiency and effectiveness of experiments with systems -- simulated or real-life" (16:259). A systematic approach to determine the most important inputs and their values is at the heart of experimental design.

Two processes called group and factor screening can provide an initial filter to help determine a potential set of important factors. Once the important factors are identified, selected patterns of the factors and factor levels are chosen to further investigate the response function relationship (4:17). The experimental design information is recorded in a table called the design matrix.

Although experimental design is an art as opposed to an exact science, it may be possible to model the process an expert uses to conduct experimental design. Expert systems capture the knowledge of an expert in a computer program

through a rule set, enabling the user of the system to mimic the decision process the expert uses (11:1). The decision support system will incorporate an expert model to support the selection of the best experimental design.

Sub-objectives.

1. Understand how an RSM expert chooses experimental designs and conducts group and factor screening.

2. Create an expert system that models the process an RSM expert uses for experimental design, group screening, and factor screening.

3. Determine which design factors are most important in the underlying more complex model.

4. Create a design matrix based on the choice of factors and levels of those factors.

Regression. "Regression analysis is a statistical tool that utilizes the relation between two or more quantitative variables so that one variable can be predicted from the other, or others" (19:23). Regression determines the relationship between the input parameters from the design matrix and the output response. The result is a model of the original model called a meta-model. The meta-model represents a simplification of the original system.

Sub-objectives.

1. Understand the mathematics of regression and include a regression model in the decision support system.

2. Regress an experimental design matrix with simulation model outputs, or responses, and obtain a meta-model of the response surface.

3. Determine the meta-model's aptness through diagnostic statistics, including residual plots, correlation matrices, and the Wilk-Shapiro statistic.

4. Determine the meta-model's lack of fit where the model equation does not properly describe the underlying simulation model.

5. Evaluate the meta-model's explanatory ability.

6. Determine the mean response and variance of the meta-model's parameters.

Simulation Models. The purpose of a simulation model is to describe, or model, the operation of a system. The system has inputs the model acts upon to generate outputs. Understanding the system and the relationship between the inputs and outputs is the primary focus of response surface methodology. A system for study with this project was generated using the simulation language SLAM II.

Sub-objectives.

1. Develop a SLAM II model for use with the decision support system. Create a Program Evaluation and Review Technique (PERT) model that calculates the expected time in system based on the activities, activity duration times, and nodes.

SCOPE

Although there are many areas of research in this effort, the focus will be on developing a kernel DSS that supports RSM analysis for six experimental design types and up to 39 factors.

II. BACKGROUND

Response surface methodology (RSM) uses statistical techniques to describe the relationship between a model's input and output variables (4:1). A decision support system (DSS) combines human judgement and computer technology to improve the process of making decisions (15:253). This research develops a decision support system to support response surface methodology analysis of a simulation model.

The three primary components of RSM are experimental design, regression and optimization. This chapter will review the three, as well as discuss decision support systems and expert systems.

Experimental Design

The roots of experimental design are in industry where setting the correct input values to a system has a direct impact on the quality of the product. Quality is and has been stressed in Japan and is now gaining greater importance in the United States. Experimental design allows for better quality and fewer defects by isolating the important factors that drive a process. Schmidt writes that "experimental design is a scientific approach which allows the researcher to better understand a process and how the inputs affect the response [output]" (24:1-2).

There are a number of reasons why experimental design is important. Varying multiple factors one at a time allows a more efficient use of limited resources and permits the calculation of interaction effects among the factors (16:269). Schmidt identifies three more reasons: "1) improved performance characteristics; 2) reduced costs; and 3) shortened product development and production time" (24:1-2). Experimental design can enhance any experiment by ensuring the correct data is collected and the best possible use of limited resources, but there are still areas to consider when using an experimental design.

Experimental Design Considerations. There are two important potential sources of equation bias, when attempting to define the correlation that may exist between the independent input factors and the dependent responses, that have a bearing on experimental design. First, it is important to have the input factors be uncorrelated among themselves to be able to concretely state that the change in the response was due directly to the input (24:1-13). Second, to minimize the error or effect of the data collection technique the experimental runs should be run in a random order. During any experiment, there will be factors out of the control of the experimenter. To minimize the effects due to the factors, Schmidt advocates randomization of the experimental runs. This means running

the experiment in a random order to average the effects of the uncontrolled variables (24:1-13). A completely orthogonal experimental design is an example of a design type that insures uncorrelated estimates of the independent factors and that no dependencies exist between the factors.

Experimental Design Types. There are many types of experimental designs to accomplish different tasks. This thesis focuses on several two and three level designs. A two level experimental design indicates each factor, or variable, can be set at two different settings. A two level design is useful in creating a first order estimation of the response surface. A three level design indicates each factor can be set at three settings. The three level designs can be used to estimate a second order meta-model of the response surface.

The two level designs used in this effort are full factorials, fractional factorials (developed by Box, Hunter, and Hunter), and Plackett-Burman designs. The three level designs used are full factorials, central composite designs, and Box-Behnken designs.

All the designs use the same coding scheme. A pattern of -1s, 0s, and 1s is placed in a matrix corresponding to low, midpoint (only for three levels), and high settings of each of the factors. The designs are created in such a way that each of the coded columns of the design matrix are

orthogonal, or nearly orthogonal, to every other coded column. Orthogonality removes the correlation between the factor coefficients, which helps differentiate between the variables in the regression analysis.

The core designs were chosen on several criteria. First, they were the design types requested by the expert. Second, they generally require the fewest number of responses to get the necessary information to generate the meta-model. Third, they are all orthogonal, or nearly orthogonal, allowing for uncorrelated estimates of factor coefficients.

Regression

Regression uses a mathematical model to define the relationship between variables. The regression relationships between the variables are sometimes quite simple, but more often the relationships are complex. (28:30). In RSM, the regression model is a reduced model of the original system or process that is called a meta-model.

The regression model is

$$\mathbf{Y = bx + e}$$

where

Y = the value of the predicted, or dependent, variable
x = the value of the predictor, or independent, variable(s) used to determine Y
b (called beta) = the constant x is multiplied by to get the value of Y
e = the random error term

Ordinary Least Squares. The method used to determine the values of the regression model is called ordinary least squares. It was originally developed by Legendre and Gauss in the early 1800s and completed by Sir Ronald Fisher in the 1920s (9:423-425).

The ordinary least squares method determines the values of the regression model by fitting a line based on the independent variable(s) x and the dependent variable Y. The line defined by the regression model is the one that has the smallest total square difference between the value of the line and the value of Y at each x. Another way to express this difference is the least squares of the deviations -- hence, the term ordinary least squares.

The process is easiest to visualize when there are only one dependent and one independent variable. When all the values of the variable x are plotted against the corresponding values of the response Y, the regression line calculated by ordinary least squares is that line which has the smallest total sum of the squared differences between the values of Y and the values of the line at each of the x values.

Assumptions and Definitions. The following are important assumptions and definitions of the linear regression model.

1. The random error term, e, has a mean $E(e) = 0$ and variance of σ^2.

2. The error terms are uncorrelated, so the covariance between the error terms $cov(e_i, e_j) = 0$.

3. The difference between the regression line and the observed value of Y at a point is $\hat{e}$.

4. Each of the values of the response variable Y come from the same probability distribution with a mean $E(Y) = bx$ and variance of σ^2.

Limitations. Although regression is a valuable tool for understanding and making inferences about a system, there are some restrictions. First, many simulation models are deterministic and do not have variance error. In this case, many regression statistics cannot be used (32:24). Second, before the regression results can be used, the assumptions must be tested. The tests are complex and expert advice is often needed. Decision support systems can be created to assist the user in determining which tests are appropriate and their meaning.

Decision Support Systems

This section concentrates on the general characteristics of a DSS and not the specifics of the man-machine interface, model base, or data base. Specifically, this section covers the topics of decision making, adaptive design, information requirements, concept mapping, and storyboards.

Decision Making. The heart of a DSS is the decision that the system supports. This thesis attempts to outline a

DSS that supports the decisions of RSM, such as which experimental design is best suited for a particular test. To understand how a DSS can effectively facilitate the decision, the decision making process must be understood. Dr. Herbert Simon describes the decision process as first, problem solving which involves information gathering, goal setting, and alternative determination, and second, the evaluation of the alternatives and the actual decision making (27:11).

How people problem solve and reach decisions is an on-going research effort, but there are certain known facts. Simon determined that

> people solve problems by selective, heuristic search through large problem spaces and large data bases, using means-ends analysis as a principal technique for guiding the search. (27:13)

Simon describes the means-ends analysis process where

> in means-ends analysis, the problem solver compares the present situation with the goal, detects a difference between them, and then searches memory for actions that are likely to reduce the difference. (27:21).

The ability to effectively reduce the amount of information reviewed and processed is one of the most important things people do when making decisions. Simon states that people cut-down problems using approximation and heuristic techniques (27:13).

Artificial Intelligence (AI) is an effort to duplicate human intelligence by using the computer's ability to

rapidly process great amounts of information. AI in the form of expert systems attempt to quantify the human's expertise through rule sets which mirror the expert. Simon observes that expert systems on computers can use brute force to examine many rules quickly and rely less on heuristics while humans must resort to heuristics to compensate for the relative lack of speed (27:22). Computers aiding the human mind can increase productivity, but as Simon points out we must first have a "better understanding of how problems can be solved and decisions made" (27:31). Because decision making is typically difficult to capture and may change, the design of a system to support those decisions should use a requirements process that can dynamically respond to the changing environment.

Adaptive Design. Given that users have inherent biases and uncertainty exists in some form, requirements will typically change as the process evolves and becomes better defined. A structure is needed that allows the dynamic change to take place within the organization.

Adaptive design attempts to account for this changing environment. The most common method for building information systems is to freeze the requirements at some point in time and build it off-site, away from the user. In contrast, Valusek makes the distinction that adaptive design "refers to design and development of the system at the

users' location and at their convenience" (30:106). Adaptive design recommends starting small then allowing the system to grow as requirements change (30:106). This concept fulfills Keen's idea of giving the users something to work with. Keen accepts the user's inability to know the requirements and advocates the building of an initial system "to give users something concrete to react to" (14:15). It is generally easier to edit than to create something.

Information Requirements. The users are responsible for determining the information requirements. There should be a systematic process to gather this data. Davis outlines

> four strategies for determining information requirements: (1) asking, (2) deriving from an existing information system, (3) synthesis from characteristics of the utilizing system, and (4) discovering from experimentation with an evolving information system. (7:12)

The second strategy represents a user's anchoring and adjustment, while the third strategy is a highly structured approach which breaks down the process into input, process, and outputs (7:14). Davis mentions the third strategy's analysis as "systematic and comprehensive" (7:18). The other two strategies are self-explanatory. The lack of user involvement in defining the system requirements can lead to incorrect or misleading results in the final product. Valusek suggests that users should become more actively involved and even initiate the process to ensure that the requirements are defined correctly (30:108).

Concept Mapping. A method for obtaining the user's requirements is the concept map. Concept mapping is a graphic brainstorm that shows the interconnections of the user's thoughts. It is free form and has no rules. In a class handout, Lt Col Valusek emphasizes that a concept map "communicates ideas", "helps the decision maker limit or bound the problem", "allow[s] the expert to brainstorm" and "gives a historical record of understanding" (31:1). The concept map describes the components of the entire process, but frequently a smaller sub-set of the concept map is used to form the core or kernel of the DSS (30:107).

Storyboards. The builder needs a set of firm requirements from which to build the DSS. One way to accomplish this is to create a storyboard that describes the computer screens and the actions that are to be accomplished. Valusek writes that the storyboard is necessary "to depict the connectivity of the entire process" (30:108). The storyboard concept is used within the movie industry to allow directors to visually see what the scenes will look like before they shoot them (1:2). Andriole points out that "they [storyboards] are also intended to consume relatively little of the systems design and development budget, while protecting that same budget from false starts, inaccurate requirements definitions, and over-eager programmers" (1:4).

The storyboard conveys the user's requirements to the builder, but the storyboard needs a set of standards to ensure a quality set of requirements. Sprague and Carlson's DSS book outlines what should be present in a DSS. When these guidelines are applied to a storyboard, it ensures it meets the standards necessary to convey ideas to a builder (30:105). Sprague and Carlson offer the following definition of the standards.

> The capabilities of the DSS from the user's point of view derive from its ability to provide representations to help conceptualize and communicate the problem or decision situation, operations to analyze and manipulate those representations, memory aids to assist the user in linking the representations and operations, and control mechanisms to handle and use the entire system. For obvious reasons, we call it the ROMC approach. (29:96)

Valusek advocates that ROMC be applied to "intelligence, design, and choice," Simon's three phases of the decision making process (30:108). Sprague and Carlson define them in the following quote.

> Intelligence. Searching the environment for conditions calling for decisions. Raw data are obtained, processed, and examined for clues that may identify problems. Design. Inventing, developing, and analyzing possible courses of action. This involves processes to understand the problem, generate solutions, and test solutions for feasibility. Choice. Selecting a particular course of action from those available. A choice is made and implemented. (29:95)

Within a decision support system, the storyboard requirements also indicate where models can be used to process information to support the decision maker. For this

system, the storyboard established that an expert model would be appropriate to assist the user in determining the best experimental design.

Expert Systems

An expert system is a "computer program which encompasses the knowledge of an expert in a domain of interest" (17:136) and allows a user to apply this knowledge to assist decision making. A key to a successful expert system is the ability to capture the entire realm of the domain-specific knowledge of interest. This was made possible by the development of computers with large memories. Even so, expert systems have been attempted only in relatively structured areas where the expert knowledge can be easily isolated and recorded. Most of the current research in expert systems involves formalized areas such as medicine, computer configuration, and geological exploration (8:28).

After the domain knowledge is captured, the rules and heuristics an expert in the field uses to make decisions is coded into a computer program. The program sorts through data looking for key information similar to the way an expert uses data. The recent successes of expert systems have been due to the hard work of the expert system designers in formalizing the experience and heuristics the decision makers use (8:29).

Dr William Gale (10:198-200) listed several other desired characteristics for an expert system, including a recognized need for help, a means of introducing the system, motivated collaborators and experts, and a task that takes a human at least several hours.

Expert Systems in Statistics. In the last decade, the topic of applying expert systems to statistics has generated great interest, including the specific area of linear regression. One of the primary reasons is that powerful generalized statistical packages are now available that do nearly all of the commonly used statistics. These packages are available to the novice and expert statistician alike. However, knowledgeable statisticians are concerned because the packages do not explain how the statistics should be used (17:134-135). In regression, seemingly "good" results can be obtained without satisfying the regression model assumptions.

Another reason for interest in statistical expert systems is that the desired characteristics listed by Dr Gale have been met. For example, the ability to capture many statistical techniques in a single package indicates there is a formalized body of knowledge. These packages allow many analysts to use statistics, but just as easily allow the analysts to misuse them. A 1979 study determined that only 28% of those in operations research departments of

large U. S. firms had a mathematical or statistical background (17: 135).

In the 1980s, several universities and laboratories researched statistical expert systems. Mellichamp (17:136) summarized the major statistical expert systems. Haaland, Yen and Liddle developed DEXTER in 1985 to assist experimental design. Porter and Lai developed STATPATH in 1983 to help identify appropriate statistical procedures based on research objectives. Mellichamp and Park developed SESSA in 1988 to address the statistical issues involved in simulation analysis. In 1984, Gale and Pregibon developed REX to assist regression analysis.

REX has direct applications to this research and will be discussed further.

<u>REX</u>. REX stands for Regression EXpert. It

> advises a user in the analysis of regression problems. It guides the analysis by testing assumptions of regression, suggesting problem transformations when assumptions are violated, and justifying its suggestions when requested. It interprets intermediate and final results, and instructs the user in statistical concepts (10:173).

Gale (10:174-189) goes on to describe how to operate REX. REX communicates with the user through a series of computer screens and windows. It begins an analysis session with personalized questions and determines the user's level of expertise. The user then provides the variables for analysis. REX automatically checks for assumption

violations. If assumptions are violated, REX provides advice on potential remedies, such as data transformations. As the analysis continues, REX interprets results for the user. When the user needs to make decisions, REX offers alternatives and advice.

A key aspect of REX is the use of graphs. Gale (10:174-189) writes that instead of providing only tables and numbers, REX also gives graphical output. For example, when REX suggests data transformations, it plots the data before and after the transformation. The graphs, plus a written explanation, help the user understand the usefulness of the transformation.

REX shows how expert systems can be applied to statistics and in particular to regression. There are also expert systems in experimental design.

Expert Systems in Experimental Design. As interest in improving quality and controlling costs of products increased, experimental design was embraced as a way to better understand a process. Experimental design is an excellent way to allocate scarce resources to determine the system factors that drive the process, but it is difficult to choose which factors to model. Currently there are no expert systems in experimental design that help someone decide which factors are important as well as which

experimental design to choose once the number of factors are known.

There are a number of software packages that provide basic experimental designs, but BBN Software Products Corporation has created a software package called RS/Discover that assists the user in creating the experimental design, once they have selected the factors they are interested in examining. This package does not offer advice about which factors are worth considering initially. Selecting the factors must be decided on an individual basis, using a person knowledgeable about that particular operation. Also defining the important factors is usually an iterative process. Once the factors of interest are designated, an experimental design can be chosen to assist in minimizing data collection runs and provide the best input, yielding the regression model with least correlated estimates of the factors.

A quote from the BBN advertising literature explains that the intent and function of their software packages is "to bring expert system technology to statistical analysis and experimental design." As the quote indicates, the purpose of their software is to provide an environment to conduct analysis and design. RS/Discover outlines the available experimental designs based on the number of responses input to the program. It may recommend a

screening design to eliminate insignificant factors, if the number of factors exceeds a certain threshold. This is one of the few times true expert advice is offered during RS/Discover. The real power resides in the good environment that the program provides to create experimental designs. However, most of its assistance is in the form of error checking and data validation.

RS/Discover joins with another program called RS/Explore to form an entire package. RS/Explore is similar to REX in that it provides a graphic interface and advice about hypothesis testing. It is not as powerful as REX, because it lacks the capability to suggest transformations or comment on assumption violations.

Optimization.

It is often of interest to find the maximum or minimum point on the meta-model surface. The meta-model surface may have either a bounded or unbounded response surface region. If the response surface is bounded, a unique optimum solution can be found directly. If the region is unbounded, the process to find the optima is more complex. The initial surface generally provides information about the direction for the next experiment to take place and new RSM equation and optimization. This unbounded process continues to examine different response regions until a bounded surface region is located.

Summary

The purpose of this research was to develop a decision support system for response surface methodology. Response surface methodology attempts to find the relationship of key inputs to the outputs of a system.

A decision support system (DSS) combines human judgement and computer technology to improve the process of making decisions (15:253). Expert systems are a subset of decision support systems. They capture the knowledge of an expert in a computer program, enabling the user of the system to mimic the decision process the expert uses (11:1).

Chapter III. Methodology

Approach

This chapter describes the steps required to build a decision support system that supports response surface methodology analysis.

Sub-objective Resolution

This section corresponds to the sub-objective section of Chapter I. It describes the effort required to accomplish each of the sub-objectives.

Decision Support Systems.

1. Objective: understand how decision support systems model a decision process.

Response: interview DSS experts and review DSS texts on the requirements for the three components of decision support systems.

2. Objective: capture the RSM process and determine the decisions.

Response: Determine which RSM experts to interview to understand the process and decision. Use concept maps to capture the knowledge of the expert. Review decision support system literature for similar applications.

3. Objective: select the part or kernel of the concept map to implement for the DSS.

Response: work with the user to select the system kernel that will serve as the baseline prototype.

_4_. Objective: determine the users requirements and create the medium to store the requirements.

Response: use storyboards to capture and covey the user's requirements for the DSS.

_5_. Objective: create the correct models to support the decisions.

Response: review the storyboard and the user's requirements to assess the need for models in the DSS.

_6_. Objective: create the correct data base to support the decisions.

Response: determine what data drives the process and determine the best method to capture the information and how it should be stored to facilitate the decision process.

_7_. Objective: develop a man-machine interface (MMI) that is user friendly and correctly displays the needed information for the user. Additionally, develop the right screens and menus that allow the users to apply the DSS efficiently.

Response: review the storyboard with the user to ensure that not only are the requirements being met but also that the MMI is correctly stated and user friendly.

_8_. Objective: provide the necessary interface to output the design matrix in the correct format for a regression package.

Response: determine what inputs are needed by the regression package and create the appropriate file.

_9_. Objective: provide the correct input interface to enter in the responses from the simulation model or other experimental output determined by the design matrix inputs.

Response: understand the interconnection between the design matrix and response vector to create the suitable interface.

Experimental Design.

_1_. Objective: understand how an RSM expert chooses experimental designs and conducts group and factor screening.

Response: interview knowledgeable RSM experts in multiple one hour question and answer sessions to secure their knowledge in a storyboard. Study experimental design and RSM to define rules for experimental design, group screening, and factor screening.

_2_. Objective: create an expert system that models the process an RSM expert uses for experimental design, group screening, and factor screening.

Response: code the expert's knowledge into a rule database that can be traversed in order to arrive at a conclusion given a set of conditions.

3. Objective: determine which design factors are most important in the underlying, more complex model.

Response: conduct group and factor screening based on the rules developed from Objective 1.

4. Objective: create a design matrix based on the choice of factors and levels of those factors.

Response: based on the user's inputs, create a design matrix. Output the results to a file.

Regression.

1. Objective: understand the mathematics of regression and incorporate a regression model in the decision support system.

Response: interview experts and review texts. Search for public domain regression software or build a regression model that can be incorporated in the DSS.

2. Objective: regress an experimental design matrix with model outputs, or responses, and obtain a meta-model of the response surface.

Response: regress the design matrix with the model responses using the regression model.

3. Objective: determine the meta-model's aptness through diagnostic statistics.

Response: run heteroscedasticity (i.e., variance) tests, normality tests, F-tests, plot residuals, and review correlation matrices to test for model aptness.

4. Objective: determine the meta-model's lack of fit where the model equation does not properly describe the underlying model.

Response: conduct lack of fit tests.

5. Objective: evaluate the meta-model's explanatory ability.

Response: assess explanatory ability through statistical techniques, including R-squared values, F tests, and P-values.

6. Objective: determine the mean response and variance of the meta-model's parameters.

Response: estimate the parameter means, standard errors, variances, and Student t test statistics, and P-values.

Simulation Models.

1. Develop a SLAM II model for use with the decision support system. The model will be a general Program Evaluation and Review Technique (PERT) model. It will calculate the expected time in system based on the activities, activity duration times, and nodes.

Response: Review books on SLAM II by A. Alan B. Pritsker, the author of the simulation language. Create the PERT model.

Delineation of Research Activities

Because this was a two person thesis, the various research activities required careful delineation and orchestration.

Most importantly, Capt Leeper and Capt Meidt had to work together to form an integrated and complete program. The intent of using the joint thesis format was to create a cohesive program that met the design requirements and worked seemlessly. Thus, even though many activities were done separately, detailed coordination was necessary to ensure the pieces fit together.

As for the specifics, Capt Meidt was responsible for creating the experimental designs given the inputs of: design type, number of factors, resolution, number of center points, and number of design repetitions. Capt Meidt was also responsible for the regression model. This entailed retrieving the experimental design information and responses, choosing the variables to regress, transforming the responses when necessary, calculating the regression statistics, and formatting the statistics for output.

Capt Leeper was responsible for the experimental design expert system and the overall implementation of the

storyboard, including the man machine interface, the operating environment and any other operating system consideration. The expert system assists the user in choosing the appropriate experimental design through choices and expert advice. The language Clipper™ provides the screen displays, the query system, and ties the different program components together. Capt Leeper was responsible for creating the screen displays and integrating the various programs into the Clipper shell.

IV. APPLICATION OF METHODOLOGY

Kernel Selection

The goal of this research effort was to create a decision support system for response surface methodology, using adaptive design. The primary decision supported in this system is choosing the best experimental design for a given situation. This chapter will outline the process of building the system.

Adaptive Design. As pointed out earlier, adaptive design is a good way to respond to a changing program environment by providing a flexible atmosphere to create the system. In William Frank Hippenmeyer's thesis on Noncombatant Evacuation Operations, he wrote that "Users do not know, or cannot articulate, what they want and need. They need an initial system to react to and improve upon" (12:3-3). The initial program provides a prototype to continue building from until the user's requirements are met. In other words, adaptive design provides a methodology for system evolution (12:3-3). One of the main reasons this design scheme was chosen was its flexibility given the knowledge that things would change. This design philosophy provided the framework necessary to accommodate the changing requirements through the use of concept maps and storyboards discussed later in this chapter.

Requirements Determination. In this case, the requirements were generated from a single user, Major Kenneth Bauer, PhD, of the Air Force Institute of Technology, because he was an acknowledged expert in the RSM field and he had to ultimately approve the program. Focusing on a single user proved to be both an advantage and a disadvantage. An advantage was that there was no need to resolve any conflicts about program requirements between users. A disadvantage was that there was no point of comparison to determine if the requirements accurately reflected the needs of other RSM users. Because of problems associated with getting information from users, it was necessary to use the tool of concept maps to graphically capture the requirements from the user.

Concept Mapping. The concept map (Figure 1) was generated during several one-hour sessions with the expert, Major Bauer. With his help, the three major areas of RSM were quickly identified. They were experimental design, regression and optimization. The concept map displayed the key ideas of RSM. Although the concept map may not have completely described every nuance of RSM, it did describe what Major Bauer felt was necessary to accomplish the core of RSM.

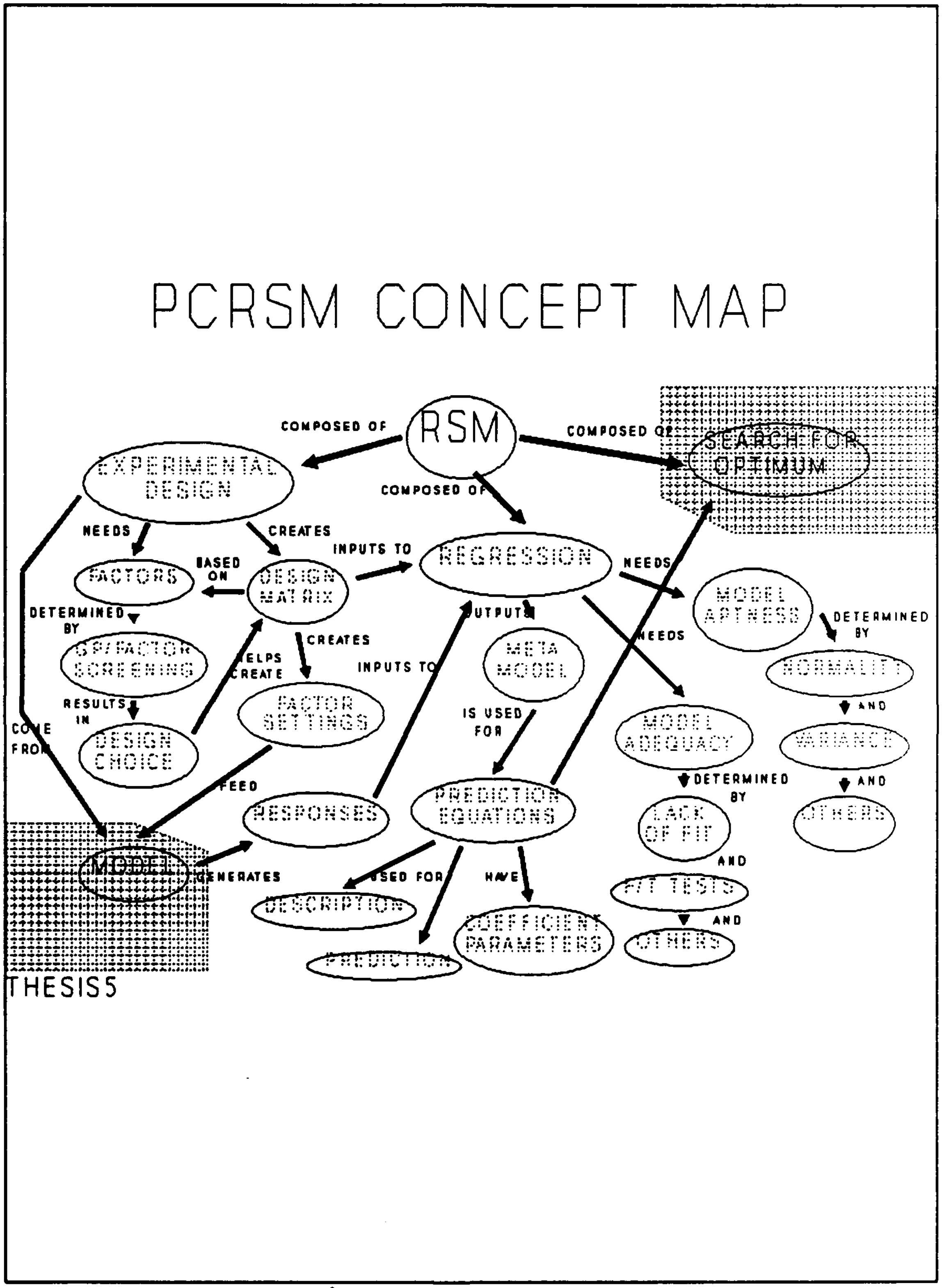


Figure 1. PCRSM Concept Map

The Kernel

The builders in concert with Major Bauer selected the kernel based on a subset of the most important requirements necessary to conduct RSM analysis. The kernel for the project consisted of experimental design and regression, excluding optimization. Major Bauer felt that these two areas were sufficient to create a useful product that could be expanded later by other thesis students. Once the kernel was selected and understood, it was necessary to determine the actual system requirements and assumptions that would be present in the system. The following requirements were a direct result of the concept map and storyboard depiction of the RSM process obtained from Major Bauer.

Overall System Requirements

1. Create a program that does RSM analysis on a personal computer of the Z-248 AT class with 640 kilo bytes of random access memory (RAM).

2. The program should be stand-alone requiring no extra files or input other than what the user provides.

3. The program should be usable by both AFIT and the Department of Defense.

Experimental Design Requirements

1. Create two-level designs with interactions, center points, and replications. The designs should include full factorial, fractional factorial, and Plackett-Burman designs for 2 to 39 factors.

2. Create three-level designs with interaction terms, center points, and replications. The designs should include full factorial, central composite and Box-Behnken designs for 2 to 6 factors.

3. Create factor screening designs.

4. Create group screening designs.

5. Factor settings.
   a. Create the low and high values for each factor.
   b. Save the inputed values.
   c. Retrieve the previously saved values.
   d. Retain significant factors found in factor screening.

6. Create raw data matrices for single and multiple group factor models.

7. Document the experimental design knowledge.

<u>Regression Requirements</u>

1. Provide a design matrix and response vector interface that can:
   a. Read in experimental designs.
   b. Read in from file or allow keyboard entry for responses.
   c. Allow user to choose specific variables to regress.
   d. Perform transformations on responses.

2. Perform least squares linear regression calculations.
   a. Calculate model and individual sum of squares.
   b. Calculate aptness statistics, including residual plots and the Wilk-Shapiro test for normality.
   c. Calculate model statistics, including F tests, Student T tests, coefficient values, standard errors, variances, covariances, and correlations.

3. Generate regression output to display on screen or send to a file or printer.
   a. Create residual plots.
   b. Create an ANOVA table.
   c. Create a coefficient table.
   c. Create a lack of fit ANOVA table.
   d. Create a variance-covariances matrix.
   e. Create a correlation matrix.
   f. Create a table with design matrix, responses, and predicted responses.
   g. Create the $(X'X)^{-1}$ Matrix.

<u>Assumptions</u>

1. The user wants to know how many runs it will take to execute an experiment given a certain experimental design. The user wants the minimum number of runs to do the experiment, because each run costs time and money.

2. The program should minimize the amount of keyboard input from the users, asking them to enter information only when absolutely necessary.

3. The user understands the terms and concepts of RSM. The user is not an expert, but someone familiar with RSM.

4. The user understands the system or simulation under study. The user can determine which factors are of interest and what ranges of values the factors can take on.

5. The user must use the included experimental designs with the regression package, because of the format structure of the design file.

6. Group and factor screening designs are always resolution three.

7. The middle value for a three-level design is the arithmetic mean between the low and high values entered from the factor settings.

8. Entire designs can be replicated, but individual design settings can not be replicated. The exception is center points, which can be individually replicated.

Storyboard. The initial storyboard was generated from the concept map and kernel to capture the actual user requirements for the process. The storyboard proved extremely helpful throughout the process by serving as a constant reminder of the user's requirements. It evolved as new information was added and a better understanding of needs developed. This points out the usefulness of the storyboard to meet the dynamic environment of user requirements. For example, the final storyboard differed from the original by collapsing two screens into one screen

and by removing unnecessary choices. These changes provided a clearer understanding of the user's requirements and a better designed program.

<u>Programming Environment</u>. A DSS requires a man-machine interface, data base and model base. From the beginning, two things were obvious. First, through Lt Col Valusek's influence, windows and cursor selection for menu options provide the best man-machine interface. Second, there would exist a data base for design choices. Because of the necessity to incorporate the two important design characteristics and the constraint that the program run on a PC, a powerful programming mini/main-frame computer environment like the Sunview windowing development library could not be used for this program.

The dBase-like language Clipper was chosen as the program language because of its capability to quickly create menu windows and its data base capability. It also interfaces with the C programming language, which the regression program and design matrix program were written in. The Clipper language proved to be extremely capable of creating the proper man-machine interface identified through the storyboards.

At the beginning of the project, it appeared that a database would be required to store each of the experimental designs. However, the database capability of Clipper was

not used because the design matrix was directly encoded within the design matrix program, instead of storing the designs within a database. The designs created by the program are sent to a flat file for use by the regression module.

The Kernel Program

This section briefly outlines the RSM kernel implemented in PCRSM. The majority of the thesis time was spent creating the program.

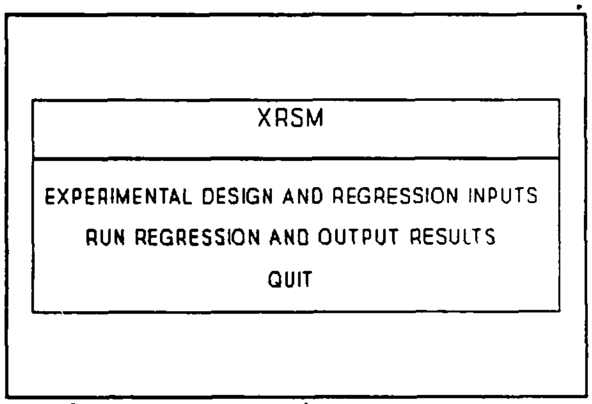


Figure 2. Main PCRSM Menu

The main menu (Figure 2) has three options: EXPERIMENTAL DESIGN AND REGRESSION INPUTS, RUN REGRESSION AND OUTPUT RESULTS, and QUIT.

Experimental Design and Regression. The RSM menu (Figure 3) controls the experimental design phase of analysis. It has four options: EXPERIMENTAL DESIGN, RUN REGRESSION, OPTIMIZATION, and EXIT. Since the optimization option is not available in the kernel program, selecting OPTIMIZATION leaves the main menu on the screen.

Selecting EXPERIMENTAL DESIGN (Figure 4) brings up a menu with four options: DESIGN CHOICES, FACTOR SETTINGS, RAW DATA INPUT, and EXIT.

The DESIGN CHOICES option offers the user the opportunity to select either two level or three level designs. It then calls to the correct sub-program to create the appropriate design. There are three two level design types available for 2 to 39 factors. The design types include full factorials, fractional factorials, and Plackett-Burman designs. There are also several three level designs available for 2 to 6 factors. The design types include full factorials, central composite designs, and Box-Behnken designs. If 12 or more factors are entered, the system sends a message to the user suggesting to try group screening.

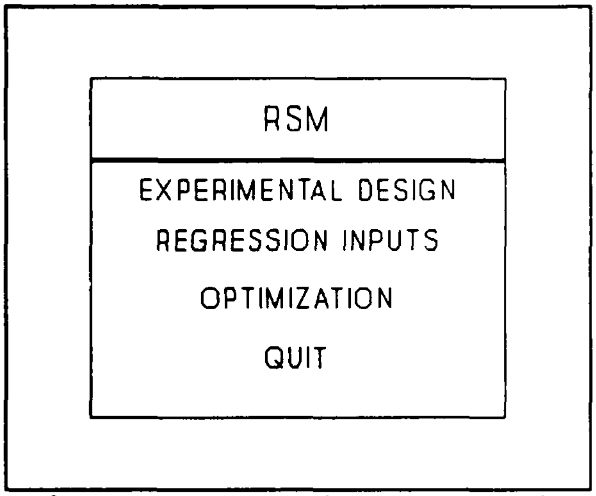


Figure 3. Experimental Design and Regression Menu

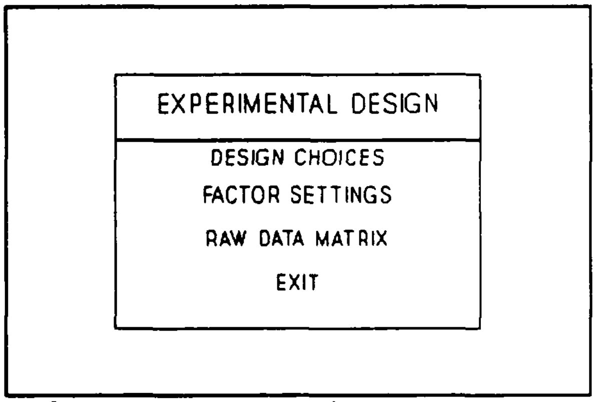


Figure 4. Experimental Design Menu

The FACTOR SETTINGS (Figure 5) option allows the user to input the low and high values for each of the factors. In addition, the user can retrieve a file and either edit it or select a subset of the factors.

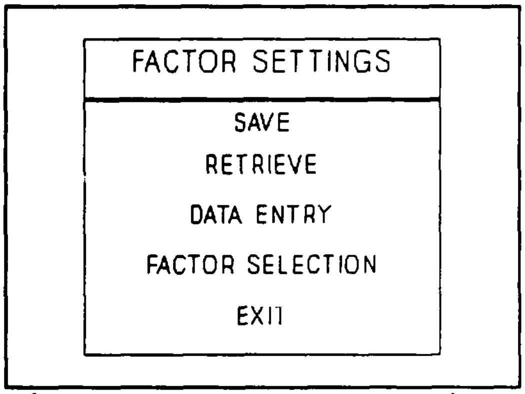


Figure 5. Factor Settings Menu

The RAW DATA INPUT option gives the user the ability to create a file for use as input to a simulation model or the settings for an experiment. The raw data matrix is a table containing the settings for each of the factors for each run. It corresponds to the coded design matrix. If group screening was selected during the design choice phase, the program prompts for the group number for each of the factors.

A detailed explanation of each of the storyboard screens and the two and three level designs is in the appendix.

The REGRESSION INPUTS menu (Figure 6) has five options. DESIGN MATRIX INPUT allows the user to input the design matrix. The design matrix program automatically generates all the interaction terms for each design. However, the user may desire a more parsimonious model with fewer terms. The VARIABLE SELECTION option allows the user to choose the

terms to leave in the model. The INPUT RESPONSES can be read in from a file or entered at the keyboard. There are times when a transformation on the responses will provide better results. The TRANSFORMATIONS option provides nine transformations. The nine options are:

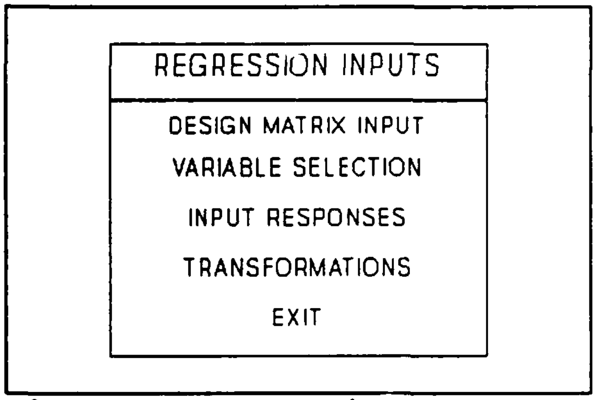


Figure 6. Regression Inputs Menu

1. power transformations (Y**alpha)
2. natural log of Y
3. log base 10 of Y
4. arcsin of the square root of Y
5. natural log of (1 + Y)/(1 - Y)
6. inverse of Y (i.e., 1/Y)
7. square root of Y
8. square of Y (i.e., Y**2)
9. natural log of (B - Y) where B is some value

<u>Regression and Output</u>. The RUN REGRESSION AND OUTPUT RESULTS option performs the regression calculations and generates tabular and graphical displays of the regression statistics. The output contains aptness and model information. The APTNESS ASSESSMENT (Figure 7) includes residual scatterplots and the

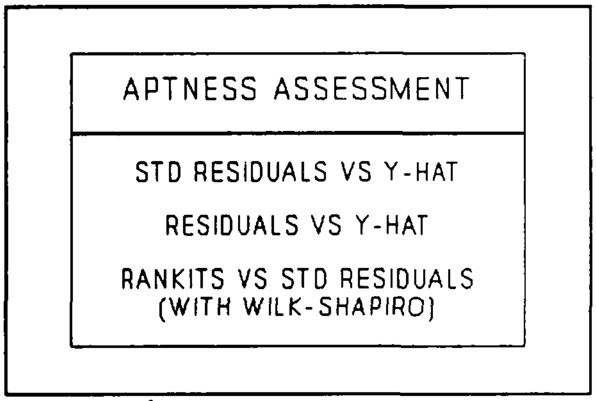


Figure 7. Aptness Assessment Menu

Wilk-Shapiro test for normality. The MODEL RESULTS (Figure 8) includes ANOVA tables, coefficient tables, and various other tables.

A detailed explanation of the regression program and the available options is provided in Appendix E.

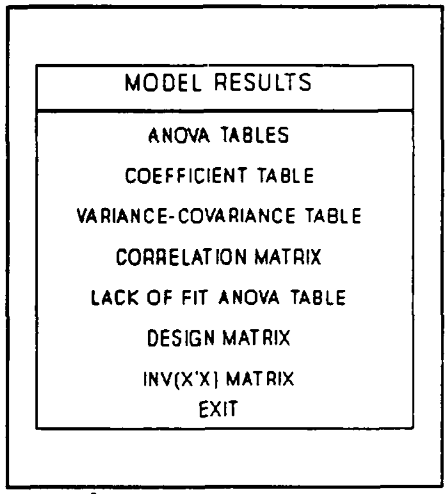


Figure 8. Model Results Menu

DSS Integration

Once the experimental design and regression program modules were constructed, it was necessary to integrate them into one package. Clipper and the C programming language provided the menu framework to integrate the programs. A problem emerged due to memory requirements for the program modules. The result was that Clipper was unable to integrate all the programs physically together. Thus, the regression and regression output programs became separate executable programs. An outside menu was created to control the entire program execution. A second difficulty occurred because the regression module requires a considerable amount of memory to perform the large matrix operations used in calculating inverses and sum of squares. Because of DOS memory limitations, the matrix size in the regression module had to be limited to a 35 by 35 matrix. Although an

inconvenience, this is not a critical problem because the program can still handle up to 35 runs (i.e., 2^5 full factorial, 31 factor Plackett-Burman, 4 factor CCD).

V. Verification and Validation

The purpose of this chapter is to provide results of the verification and validation testing of the project. The verification process determined if the individual program components performed as expected. The methodology was to use seven test cases to exercise the various components of the DSS. The validation section determined if the overall program did what was intended, support RSM analysis. The validation testing consisted of two parts. The first was an end-to-end problem analysis using the PERT model. The second was a comparison to the BBN software package RS/Discover.

Test Cases

Seven test cases were generated to support the verification and validation process. The cases came either from textbooks or were computed on Statistics II to ensure the "truth" was known.

The cases were:

1. a 2^3 full factorial design from Box (4:108-113). It studied the behavior of worsted yarn under cycles of repeated loading.

2. a 2^{5-1} Resolution V fractional design from Box (4:184-186). It studied the process used in the manufacturing of a drug.

3. a 30 factor Plackett-Burman design. The responses were generated randomly.

4. a 3^3 full factorial design from Box (4:206-214). It was a continuation of Case 1.

5. a 3 factor central composite design (4:307-312). The experiment explored the conditions leading to maximal elasticity for a certain polymer.

6. a 3 factor central composite design with 6 replicated center points (18:78-82). It investigated the seal strength of a breadwrapper stock.

7. a 4 factor Box-Behnken design (4:223).

Verification

Verification tested the ability of the program to create correct experimental designs, perform transformations on the responses, regress experimental designs against the responses, and calculate regression statistics.

Create Experimental Designs. Given the inputs of number of factors, design type, resolution, center points, and replications, the design matrices were generated for the seven examples. Each design was correct. Table 1 contains the 2^3 design matrix.

	X0	X1	X2	X3	X12	X13	X23	X123
1	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
3	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00	1.00
4	1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00
5	1.00	-1.00	-1.00	1.00	1.00	-1.00	-1.00	1.00
6	1.00	1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00
7	1.00	-1.00	1.00	1.00	-1.00	-1.00	1.00	-1.00
8	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

DESIGN MATRIX

	Y	Y-HAT
1	2.83	2.83
2	3.56	3.56
3	2.23	2.23
4	3.06	3.06
5	2.47	2.47
6	3.30	3.30
7	1.95	1.95
8	2.56	2.56

Table 1. Case 1 Design Matrix

Response Transformations. The DSS provides nine transformation options. As an example, the 3^3 design required a logarithmic transformation on the responses. Table 1 contains the transformed data.

Regression. Each design matrix was regressed with the corresponding response vector. The time for regression on an IBM PS 2 varied from a couple of seconds on the 2^3

example to 45 seconds for the 30 factor Plackett-Burman design.

Regression Results. The DSS provides graphical output for aptness tests and tabular output for model statistics.

Aptness. Variance and normality tests are produced as residual plots. Statistics II provides similar plots. As an example, the Box-Behnken design residual plots were compared to the Statistics II plots. The plots were identical. The Wilk-Shapiro test statistic for normality differed by .07.

Model. The program generates seven output tables. Each output table was generated for the seven examples and compared to the textbook or Statistics II results. In all cases, the results were similar. For example, Table 2 is the ANOVA table for the 2^3 example.

ANALYSIS OF VARIANCE

WARNING ! STATISTICAL TESTS ARE INVALID. SSE = 0 WITH df = 0. THEY HAVE BEEN ARBITRARILY SET TO ONE.

SOURCE	SS	df	MS	F
Regression	2.086	7	0.298	0.30
X1	1.125	1	1.125	1.12
X2	0.696	1	0.696	0.70
X3	0.245	1	0.245	0.24
X13	0.013	1	0.013	0.01
X12	0.003	1	0.003	0.00
X123	0.002	1	0.002	0.00
X23	0.002	1	0.002	0.00
Error	1.000	1	1.000	
Total	2.086	7		

ORTHOGONAL DESIGN

MODEL F VALUE = 0.298 P-VALUE = 0.8904
R SQUARED = 1.000
ADJUSTED R SQUARED =-2.356

Table 2. Case 1 ANOVA Table

COEFFICIENT TABLE					
VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	85.904	0.625	0.391	137.361	0.0000
X4	-3.675	0.938	0.880	-3.918	0.0020
X22	-4.329	1.407	1.980	-3.077	0.0096
X34	-4.250	1.625	2.640	-2.616	0.0225
X13	-3.825	1.625	2.640	-2.354	0.0363
X2	-1.958	0.938	0.880	-2.088	0.0587
X1	1.933	0.938	0.880	2.061	0.0615
X44	-2.579	1.407	1.980	-1.833	0.0915
X24	-2.625	1.625	2.640	-1.616	0.1318
X33	-2.242	1.407	1.980	-1.593	0.1393
X3	1.133	0.938	0.880	1.208	0.2521
X23	-1.675	1.625	2.640	-1.031	0.3246
X12	-1.675	1.625	2.640	-1.031	0.3246
X11	-1.417	1.407	1.980	-1.007	0.3356
X14	0.950	1.625	2.640	0.585	0.5707

Table 3. Case 7 Coefficient Table

Table 7 is the Coefficient table for the Box-Behnken example. It compares closely to the similar table created using Statistics II. In some of the test cases, the model coefficients differed between the book solution and the DSS. The reason for the differences is that the DSS automatically creates orthogonality between the mean and quadratic terms for three level designs by subtracting the average of the quadratic column in the design matrix (2:7-6). The benefit of the orthogonal transformation is that it removes correlation between the factor estimates. Many of the textbook examples ignored the dependency between the

quadratic terms and the mean. When the test cases were run without the benefit of the orthogonal transformation, the results were equivalent.

The Variance-Covariance Matrix for the Case 6 CCD example is located in Table 4. It shows the quadratic terms are uncorrelated with the mean, but the quadratic terms are correlated with each other. The replicated center points caused the slightly non-orthogonal design.

The Case 6 Correlation Matrix is in Table 5. It correctly compared to the Statistics II output.

Table 6 is the Lack of Fit ANOVA Table for the Case 6 example. The table agrees with Myers' results.

VARIANCE-COVARIANCE MATRIX

	X0	X1	X2	X3	X11	X22	X33
X0	0.065	0.000	0.000	0.000	-0.000	-0.000	-0.000
X1	0.000	0.094	0.000	0.000	-0.000	-0.000	-0.000
X2	0.000	0.000	0.094	0.000	-0.000	-0.000	-0.000
X3	0.000	0.000	0.000	0.094	-0.000	-0.000	-0.000
X11	-0.000	-0.000	-0.000	-0.000	0.090	0.009	0.009
X22	-0.000	-0.000	-0.000	-0.000	0.009	0.090	0.009
X33	-0.000	-0.000	-0.000	-0.000	0.009	0.009	0.090
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X123	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X12	X13	X23	X123
X0	0.000	0.000	0.000	0.000
X1	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.000
X11	0.000	0.000	0.000	0.000
X22	0.000	0.000	0.000	0.000
X33	0.000	0.000	0.000	0.000
X12	0.161	0.000	0.000	0.000
X13	0.000	0.161	0.000	0.000
X23	0.000	0.000	0.161	0.000
X123	0.000	0.000	0.000	0.161

Table 4. Case 6 Variance-Covariance Matrix

CORRELATION MATRIX

	X0	X1	X2	X3	X11	X22	X33	X12
X0	1.000	0.000	0.000	0.000	-0.000	-0.000	-0.000	0.000
X1	0.000	1.000	0.000	0.000	-0.000	-0.000	-0.000	0.000
X2	0.000	0.000	1.000	0.000	-0.000	-0.000	-0.000	0.000
X3	0.000	0.000	0.000	1.000	-0.000	-0.000	-0.000	0.000
X11	-0.000	-0.000	-0.000	-0.000	1.000	0.099	0.099	0.000
X22	-0.000	-0.000	-0.000	-0.000	0.099	1.000	0.099	0.000
X33	-0.000	-0.000	-0.000	-0.000	0.099	0.099	1.000	0.000
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X123	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X13	X23	X123
X0	0.000	0.000	0.000
X1	0.000	0.000	0.000
X2	0.000	0.000	0.000
X3	0.000	0.000	0.000
X11	0.000	0.000	0.000
X22	0.000	0.000	0.000
X33	0.000	0.000	0.000
X12	0.000	0.000	0.000
X13	1.000	0.000	0.000
X23	0.000	1.000	0.000
X123	0.000	0.000	1.000

Table 5. Case 6 Correlation Matrix

ANALYSIS OF VARIANCE WITH LACK OF FIT

SOURCE	SS	df	MS	F
Regression	70.311	9	7.812	6.59
Error	11.859	10	1.186	
Lack of Fit	6.899	5	1.380	1.39
Pure Error	4.960	5	0.992	
Total	82.170	19		

MODEL F VALUE = 6.587 P-VALUE = 0.0033
LACK OF FIT F VALUE = 1.391 P-VALUE = 0.3603
R SQUARED = 0.856
ADJUSTED R SQUARED = 0.856

Table 6. Case 6 Lack of Fit ANOVA Table

INV(X'X) MATRIX

	X0	X1	X2	X3	X11	X22	X33	X12
X0	0.037	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.056	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.056	0.000	-0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.056	0.000	0.000	0.000	0.000
X11	0.000	0.000	-0.000	0.000	0.167	-0.000	-0.000	0.000
X22	0.000	0.000	0.000	0.000	-0.000	0.167	0.000	0.000
X33	0.000	0.000	0.000	0.000	-0.000	0.000	0.167	0.000
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.083
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X123	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X13	X23	X123
X0	0.000	0.000	0.000
X1	0.000	0.000	0.000
X2	0.000	0.000	0.000
X3	0.000	0.000	0.000
X11	0.000	0.000	0.000
X22	0.000	0.000	0.000
X33	0.000	0.000	0.000
X12	0.000	0.000	0.000
X13	0.083	0.000	0.000
X23	0.000	0.083	0.000
X123	0.000	0.000	0.125

Table 7. Case 4 INV(X'X) Matrix

The $(X'X)^{-1}$ matrix for the 3^3 example is in Table 7. As discussed in Appendix E, many of the designs do not require a classical Gaussian elimination inversion, but can be

calculated directly. It was necessary to invert this particular case. The inversion was correct.

Validation

The verification process demonstrated that each of the program components individually performed as intended. The purpose of the validation phase was to determine if the DSS would support the end-to-end formulation and analysis of a problem. The SLAM PERT model was used as the example.

Repair and Retrofit PERT Problem. The PERT network chosen for analysis is Example 7-1 in the Pritsker SLAM textbook (21:217) It's a model of a repair and retrofit project. The project shown in Figure 9 consists of nine activities that are all assumed to be triangularly distributed. The activities of the pert network correspond to the RSM factors (inputs) and the expected project completion time corresponds to the response (output).

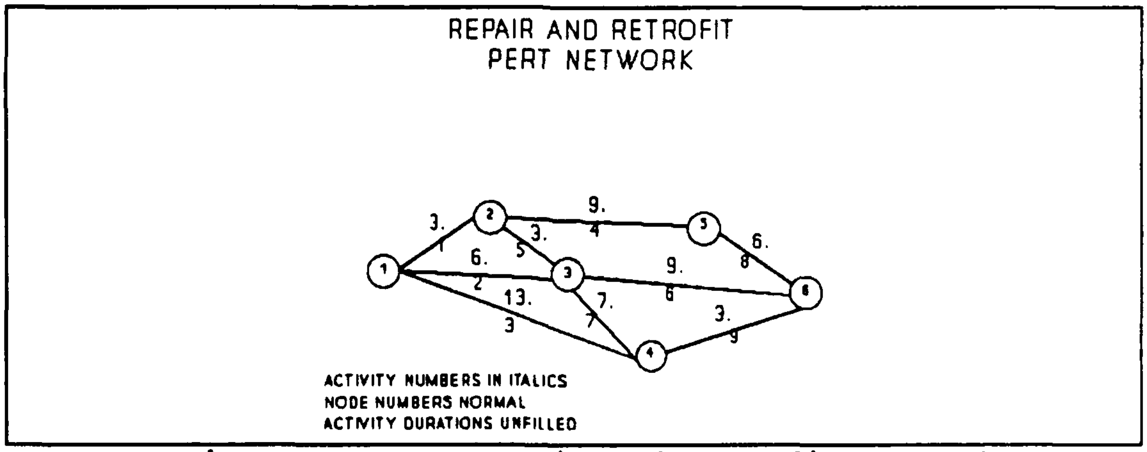


Figure 9. PERT Repair and Retrofit Network

REPAIR AND RETROFIT
PERT NETWORK

ACTIVITIES	LOW	MEAN	HIGH
1 DISASSEMBLE UNITS	1	3	5
2 PUT IN NEW ASSEMBLY	3	6	9
3 PREPARE FOR RETRO	10	13	19
4 CLEAN AND REPAIR	3	9	12
5 CALIBRATE INSTRUM.	1	3	8
6 CHECK INTERFACES	8	9	16
7 CHECK ASSEMBLY	4	7	13
8 ASSEMBLE UNITS	3	6	9
9 RETROFIT CHECK	1	3	8

ALL HAVE TRIANGULAR
DISTRIBUTIONS

Figure 10. PERT Network Activity Durations

A primary goal of this RSM analysis was to find the most important factors of a network (i.e., those on the critical path). If the most important factors existed and were isolated, it could be possible to develop a parsimonious linear model to describe the entire network with just a few parameters. However, if each of the factors were equally important, the RSM model may require all the activities. Figure 10 has the activities and associated expected minimum, mode, and maximum times. SLAM calculated the expected project completion time by generating random activity times in the appropriate distribution for each of the nine activities. This was repeated for 400 runs. The expected project completion time was the average completion time of the 400 runs, in this case 20.7.

Experimental Design and Regression. The analysis was conducted by examining the responses after changing the

expected durations of the original activities by +/- 10%. To find the important factors that may be driving the model, a $2^{(9-5)}$ resolution III design was chosen to identify the significant main effects. This required 16 runs.

In the initial analysis, all main factors and interactions generated by the DSS were left in the model. This resulted in an equal number of variables and runs resulting in a model without error. However, examination of the coefficients indicated some of the interactions term could be removed. Thus, the bottom four interaction terms of Table 8 were removed and the regression re-run.

COEFFICIENT TABLE

WARNING ! STATISTICAL TESTS ARE INVALID. SSE = 0 WITH df = 0. THEY HAVE BEEN ARBITRARILY SET TO ONE.

VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	21.006	0.250	0.062	84.025	0.0076
X7	0.394	0.250	0.062	1.575	0.3606
X6	0.281	0.250	0.062	1.125	0.4632
X3	0.281	0.250	0.062	1.125	0.4632
X9	0.269	0.250	0.062	1.075	0.4776
X5	0.194	0.250	0.062	0.775	0.5797
X2	0.181	0.250	0.062	0.725	0.6001
X1	0.169	0.250	0.062	0.675	0.6215
X4	0.131	0.250	0.062	0.525	0.6918
X8	0.119	0.250	0.062	0.475	0.7173
X13	-0.081	0.250	0.062	-0.325	0.7997
X12	-0.081	0.250	0.062	-0.325	0.7997
X23	-0.069	0.250	0.062	-0.275	0.8289
X24	-0.044	0.250	0.062	-0.175	0.8896
X14	-0.031	0.250	0.062	-0.125	0.9207
X34	-0.019	0.250	0.062	-0.075	0.9523

Table 8. Initial PERT Coefficient Table

The result of the second regression run was that all main factors were significant at the 10% alpha value level. The model looked adequate, with good residual plots and a high a Wilk-Shapiro of .941. Tables 9 and 10 contain the ANOVA Table and Coefficient Table for the model.

ANALYSIS OF VARIANCE				
SOURCE	SS	df	MS	F
Regression	8.461	11	0.769	23.96
X8	2.481	1	2.481	77.27
X13	1.266	1	1.266	39.42
X3	1.266	1	1.266	39.42
X12	1.156	1	1.156	36.00
X1	0.601	1	0.601	18.71
X6	0.526	1	0.526	16.37
X7	0.456	1	0.456	14.19
X9	0.276	1	0.276	8.59
X4	0.226	1	0.226	7.03
X2	0.106	1	0.106	3.29
X5	0.106	1	0.106	3.29
Error	0.128	4	0.032	
Total	8.589	15		

ORTHOGONAL DESIGN

MODEL F VALUE = 23.958 P-VALUE = 0.0038
R SQUARED = 0.985
ADJUSTED R SQUARED = 0.955

Table 9. PERT ANOVA Table

COEFFICIENT TABLE					
VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	21.006	0.045	0.002	468.949	0.0000
X7	0.394	0.045	0.002	8.790	0.0009
X6	0.281	0.045	0.002	6.279	0.0033
X3	0.281	0.045	0.002	6.279	0.0033
X9	0.269	0.045	0.002	6.000	0.0039
X5	0.194	0.045	0.002	4.325	0.0124
X2	0.181	0.045	0.002	4.046	0.0155
X1	0.169	0.045	0.002	3.767	0.0197
X4	0.131	0.045	0.002	2.930	0.0429
X8	0.119	0.045	0.002	2.651	0.0570
X13	-0.081	0.045	0.002	-1.814	0.1441
X12	-0.081	0.045	0.002	-1.814	0.1441

Table 10. PERT Coefficient Table

Although the residual plots did not indicate higher order terms were required, four center points were added to get information about the center of the design as well as lack of fit information of the model. Since the design was orthogonal, the only difference in the residual plots between the initial design without center points versus with center points turned out to be points in the middle of the plot. The normality of the errors slightly decreased to a Wilk-Shapiro of .930. The center points pointed out that a quadratic term was unnecessary. The Coefficient Table (Table 11) indicates that the main factors are significant at 10% alpha and the two interaction terms less significant.

However, after creating another model without the interaction terms, it was decided to leave the terms in the model. The residual plots and Wilk-Shapiro statistic looked significantly worse in the other model. The Lack of Fit ANOVA Table in Table 12 indicates the model is significant and there is no lack of fit in the model.

COEFFICIENT TABLE					
VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	20.980	0.048	0.002	435.636	0.0000
X7	0.394	0.054	0.003	7.313	0.0000
X6	0.281	0.054	0.003	5.223	0.0008
X3	0.281	0.054	0.003	5.223	0.0008
X9	0.269	0.054	0.003	4.991	0.0011
X5	0.194	0.054	0.003	3.598	0.0070
X2	0.181	0.054	0.003	3.366	0.0098
X1	0.169	0.054	0.003	3.134	0.0139
X4	0.131	0.054	0.003	2.438	0.0407
X8	0.119	0.054	0.003	2.205	0.0584
X13	-0.081	0.054	0.003	-1.509	0.1704
X12	-0.081	0.054	0.003	-1.509	0.1704

Table 11. PERT Coefficient Table with Center Points

ANALYSIS OF VARIANCE WITH LACK OF FIT				
SOURCE	SS	df	MS	F
Regression	8.462	11	0.769	16.58
Error	0.371	8	0.046	
Lack of Fit	0.184	5	0.037	0.59
Pure Error	0.188	3	0.063	
Total	8.833	19		

ORTHOGONAL DESIGN

MODEL F VALUE =	16.583	P-VALUE =	0.0002
LACK OF FIT F VALUE =	0.587	P-VALUE =	0.7195
R SQUARED =	0.958		
ADJUSTED R SQUARED =	0.916		

Table 12. PERT Lack of Fit ANOVA Table

Insights. The estimate of the mean for the model 20.9 was very close to the estimate of the completion time for the original model (20.7). This indicates that the changes caused by the +/- 10% variations in the factor levels are in a plane rather than a quadratic or other form. This also indicates that the coefficients in the model simply move the response around the plane. The fact that a first order model with interactions adequately describes the network supports this.

The main effects of the model are all positive and the interaction effects of the model are all negative. This makes intuitive sense. In general, an increase in a factor

level tends to increase the expected project completion time. However, an increase in two factor levels will not necessarily increase the system time by the full weight of both factors. The reason is that the two factors could be on parallel paths, each competing to be on the critical path. Only the factor that makes the critical path is used in the expected project completion time. Project time still increases, but not as much as if both of the factors are on the critical path.

Although the interaction effects in the model are confounded, it is believed they are representative of the activities that are closely competing for the critical path.

Conclusions. The results of this analysis indicate the final model is useful over the region investigated, which is +/- 10% of the original network. More importantly, the DSS successfully supported the RSM analysis.

BBN Software Comparison. Due to scheduling conflicts, the review of BBN's software package occurred after completion of this effort. As a result, this review provided a secondary validation of the project design and the fulfillment of requirements necessary to conduct RSM.

At the highest level, the important design requirements as outlined in the concept map were the same in both systems. There were both similarities and differences

between the systems. Two interesting similarities were the implementation of expert advice for screening designs and the number of runs recommended for a design. Both programs recommend using a factor screening design once the number of factors reach a certain threshold value (3:1-9). Also, both programs automatically create designs with the minimum number of runs for a given design (3:1-8). The differences that exist between the thesis program and the full BBN program are a result from the choices made in selecting the kernel, such as the decision to leave out optimization and the three-dimensional graphing of response surfaces. However, most of the requirements outlined in the concept map were the same as BBN.

The thesis program does things that the BBN program does not do. For instance, the BBN package does not perform transformations on the responses or provide group screening advice. Although the BBN program was powerful and did many things, it also lacked a good man-machine interface (MMI) which is critical to a good DSS.

During interviews with people who were using the BBN system, they made frequent mistakes and forgot where they were in the system. These types of errors indicate a deficient MMI. RS/Discover does not use cursor select menus or layered memory aids to assist the user. In contrast, the

same users commented on the high quality and the ease of use of the thesis program's MMI.

The goal of the thesis was not to replicate BBN software package. In fact, when the project began the designers had no knowledge of the BBN system. However, the comparison offered an independent validation and provided proof of concept.

Chapter VI. Conclusions and Recommendations

This chapter documents the observations, recommendations for further research, and conclusions from this effort.

Observations and Applications

1. A joint thesis is an excellent way to accomplish a large project in a single thesis. However, for a joint thesis to work, the project must be modular so parts can be worked on in parallel and both people must be able to work together.

2. A joint thesis probably generates a better product than follow-on theses. The reason is that much of the effort in creating a working DSS from the original concept map and storyboard involves creative problem solving. Problem solving is often easier and more effective when two people discuss the situation than one alone.

3. User's requirements change. It appeared that the better the builders understood the process and created the corresponding prototype system, the more accurately and quickly they could translate the user's requirements. This better understanding led to the evolution of the user's requirements as the user wrestled with the builder's questions about system requirements. This verified and validated the need for adaptive design and its iterative

approach to creating a system where the true understanding of the process comes only after trying to capture it.

4. In an ideal adaptive environment world, a user would define requirements directly in the form of a storyboard and give them to a builder. This project proved that this can be done, but it increased the responsibility of both the user and builder. To be effective, the user had to be more actively involved in the requirements process and the builder had to iteratively show the user the prototype. In addition, the builder had to not only be responsible for creating the program but also ensuring a correct design by actively questioning the user to gain a better understanding of the requirements. This dual role was effective because the builder invested the extra time to adequately understand the true user requirements both during the design phase of the project and throughout the entire process.

5. In general, the storyboard accurately reflected the user's requirements and had the capacity to change. As the project progressed, the original picture of the process depicted in the storyboard turned out to be slightly different. A number of features and screens were combined or eliminated. The storyboard, however, always provided a reference and point of understanding between the builder and the user.

6. The concept map is an excellent method for capturing the knowledge of the user in an accurate and timely manner. In just three concept mapping sessions with Maj Bauer, he was able to communicate his perspective and understanding of RSM into the concept map. Once the knowledge was available, it was an easy step to translate the user's information to requirements in the form of storyboards.

7. Integration between different code modules created in parallel always takes longer than is scheduled to. This was especially true in this project because of the need to integrate Clipper with the C programming language and between two different people's code.

8. Clipper's advantage of man-machine interface menus and windows may be out-weighed by the size of the program it creates. Clipper, by its nature and structure, requires a great deal of memory. This is a fixed constant and unfortunately has made an otherwise excellent language difficult to use for large projects where memory is a consideration.

Recommendations for Follow-On Research

The following recommendations are broken into four categories: overall system, experimental design, regression, and optimization. Within each category, they are listed in order of priority.

Overall System

1. Rewrite the program exclusively in the C programming language or another dBase-like language compiler that can create smaller executable files. This will allow the regression program to be fully integrated in one program for the DSS.

2. Upgrade the help option, or analysis advice, to create a true teaching environment which could be used by the expert or novice alike.

Experimental Design

1. Include an expert advice toggle that gives users experimental design information based what they are doing in the program and the level of expertise of the user.

2. Create an option for fold-over designs.

3. Allow for replications at every design point instead of the entire design only.

4. If possible, expand the two level fractional orthogonal designs beyond the current 11 factors.

5. Create three-level central composite and Box-Behnken designs for more than the current six factors.

6. Create two level full factorial designs for up to 7 factors, or 128 runs. Even if the included regression package cannot handle it, regression packages like SAS can do the calculations.

7. Allow the users to create the blocking/aliasing structure and to be able to de-alias the factors of interest. This would be possible by allowing users to select which factors they want aliased with other factors. Currently, the program creates an aliasing pattern assuming that the factors of interest should be aliased with the highest interactions available.

8. Create D-optimal designs. There are occasions where classic RSM design are not possible because of run constraints or blocking considerations. D-optimal designs attempt to create the best nearly orthogonal designs, using the determinant of the covariance matrix as the optimality criteria.

9. Create Taguchi designs for determining variance so that a design can minimize the expected variance of the product.

10. Create a random order for the experimental runs.

11. Reuse data from one design phase to the next. Determine which runs were present from the last design and therefore unnecessary to run again in the next design.

12. Offer expert advice about the number of groups given the number of factors.

Regression

1. Create ability to print or send to a file the scatterplots.

2. Create an expert system to assist regression analysis.

3. Increase size of design matrices the regression program can handle up to 128 rows.

4. Provide a graphics program that would show the response surface and optimum, if it existed.

5. Rewrite the code that does matrix manipulations to increase speed of regression calculations.

6. Stepwise regression and print out Cp statistic.

7. Perform multivariate regression.

Optimization

1. Find the minimum or maximum on the response surface defined by the meta-model. Do this for both the bounded and unbounded case.

Summary

In summary, this effort achieved its objective. PCRSM is a stand alone PC-based decision support system that assists the RSM analysis process. What started as a dream and a concept map has become a viable analysis tool. If properly managed, PCRSM has the potential for widespread use not only at AFIT, but throughout the RSM analysis community.

Appendix A: PCRSM USER'S GUIDE

INTRODUCTION

PCRSM is a personal computer based decision support system for response surface methodology (RSM). The package is designed to help someone familiar with RSM to perform RSM analysis on a PC.

PCRSM conducts group screening, factor screening, creates the coded experimental design matrices, creates raw data matrices in the original variable settings, and performs least squares regression on the design matrix and responses.

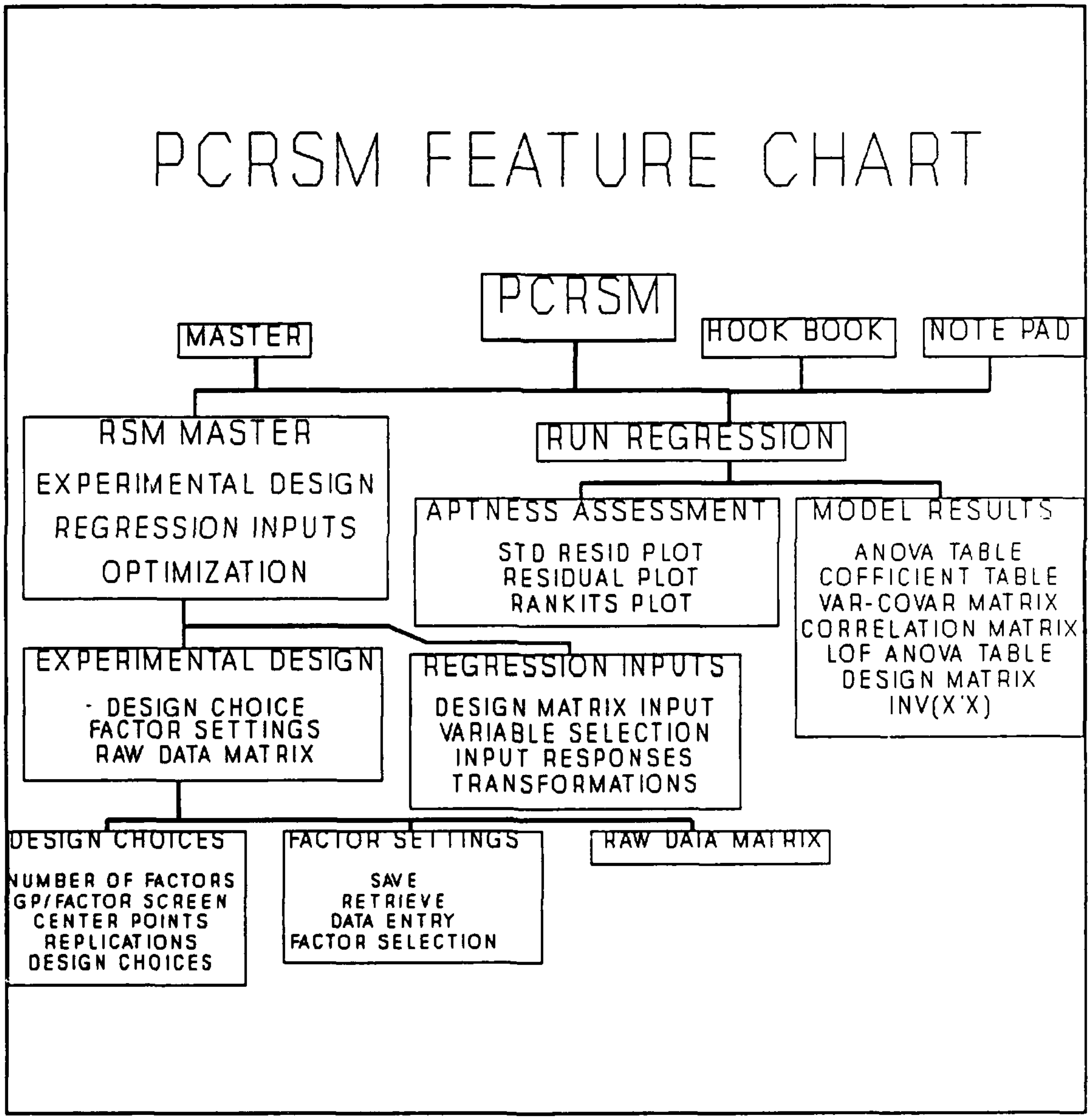
PCRSM contains six two and three level design types, including full factorials, fractional factorials, Plackett-Burman designs, central composite designs, and Box-Behnken designs. In all, there are 60 experimental designs for 2 to 39 factors.

The regression package performs model variable selection, performs transformation on the responses, provides a graphical model aptness assessment, and generates a variety of model results. The model results include two ANOVA tables, a coefficient table, the variance-covariance matrix, the correlation matrix, and the design matrix.

The user's guide is broken into three sections. The first section discusses the overall system operation and

getting started. The second section describes the experimental design process. The third section describes the regression package.

The following Feature Chart provides the menus of PCRSM for use as a reference while using this guide.



SECTION 1: PCRSM OVERALL OPERATION

PCRSM was written for use on an IBM compatible PC. The package requires that ANSI.SYS be present in the CONFIG.SYS file on your computer. If it isn't, add DEVICE = ANSI.SYS to your configuration file and reboot. Please remember to ensure that ANSI.SYS is available on the computer's path.

PCRSM uses five executable files. They are PCRSM.EXE, MASTER.EXE, REGRESS.EXE, REGOUT.EXE, AND CLIPRAW.EXE. If you have them, you are ready to begin.

Type PCRSM to begin the program. The first screen should welcome you to PCRSM and provide points of contact for comments and questions about the program.

The second screen provides the choices of conducting experimental design, running the regression package, or exiting. Highlight the choice you want by moving the up and down arrow keys and hit <ENTER>.

SECTION 2: EXPERIMENTAL DESIGN PROCESS

Choosing EXPERIMENTAL DESIGN AND REGRESSION INPUTS from the main menu begins the experimental design process. In it you can conduct group screening, factor screening, create the coded experimental design matrices, create raw data matrices in the original variable settings, select the variables in the design matrix to regress, and perform transformations on the responses.

A. F1: THE HELP FUNCTION

Pressing the F1 key from the RSM Master Menu provides help. It includes definitions of key RSM terms and concepts as well as specific PCRSM options.

B. Esc: ESCAPE

When stuck in an undesirable situation, use the escape key to back out of the immediate process.

C. F10: MASTER, HOOK BOOK, NOTEPAD

The F10 key provides three options: Master, Hook book, and Notepad. Each option can be chosen by cursor selection or typing the first letter of the word.

The master option allows you to return to the RSM Master menu from any other menu.

The hook book is intended to be used as a method to capture system development and improvement thoughts. If a system improvement is thought of at any time while in the analysis process, the idea can be immediately captured in the hook book without disturbing the analysis itself. The hook book entries can be later reviewed, edited, or deleted.

The intent of the Notepad is to capture <u>any</u> thought, analysis related or not, for later review, edit, or deletion.

D. EXPERIMENTAL DESIGN

The experimental design choice generates three options: designs choices, factor settings, and raw data matrix.

D1. Design Choices

Design choices determines and creates the experimental design based on the number of factors of interest, number of desired center points, number of desired design replications, and whether it's group or factor screening.

PCRSM contains six two and three level design types, including full factorials, fractional factorials, Plackett-Burman designs, central composite designs, and Box-Behnken designs. In all, there are 60 experimental designs for 2 to 39 factors.

When asked for the number of factors, if you want a two level design, enter the number of factors. The next screen will give the available two level design options. If you want a three level design instead, enter a 0 for the number of factors and the next screen will give you the three level design options. You will then be asked a second time for the number of factors.

The Group screening option is only available with two level designs and will automatically generate the smallest (i.e., least number of runs) Resolution III design based on the number of groups.

After the design is created, it can be saved. An extension .DES (for DESign matrix) will be added to the filename. Even if it isn't saved, the design matrix will remain the current design available for further use.

D2. Factor Settings

The uncoded, or original, values of the factors can be entered, edited, and saved in the factor settings menu.

The data entry option reviews the experimental design to determine the number of factors. Enter the uncoded low and high values of the variables. If it's a three level design, the medium value will be calculated as the arithmetic mean of the high and low.

The variables and the variable settings can also be saved. An extension .FAC (for FACtor settings) will be added to the filename.

If you want to remove variables, choose the factor selection option. Select or highlight the variables you want to keep. If variables are removed, a new .FAC file can be created and saved.

D3. Raw Data Matrix

The raw data matrix contains the variables and the uncoded settings of the variables required to generate the design points in the experimental design. It corresponds to the coded design matrix. To create the raw data matrix, get the experimental design matrix and corresponding factor settings file. The program will automatically create a raw data matrix. If saved, the extension .RAW will automatically be added.

Since the .RAW file is an ASCII file, it can easily be edited and possibly even used to directly to run your model.

E. REGRESSION INPUTS

The regression inputs menu readies the design matrix and responses for the regression package. When you've entered the design matrix and responses, exit out to the PCRSM main menu to run the regression model.

E1. Design Matrix Input

The regression model needs to know which design matrix to regress. If an experimental design was created previously in the same session, it does not need to be retrieved again. It will be used by default.

E2. Variable Selection

The experimental designs generated by PCRSM automatically create all possible interaction terms, up to as many terms as rows in the design matrix. However, you may desire a more parsimonious model based on the results of a previous regression run or prior knowledge. The variable selection option allows you to choose the terms to leave in the model.

You can save the new design matrix if you want. But even if you don't save it, the regression model will use the new model unless a new design matrix is retrieved through the design matrix input option.

E3. Input Responses

The responses, or outputs, corresponding to the design matrix can be entered at the keyboard or read in from an ASCII file. However, before the responses can be entered at

the keyboard, a design matrix must be retrieved so the total number of responses is known.

The format of the ASCII file must be as follows. The first entry on the first line must be the total number of responses. The responses should follow, one per line.

REMEMBER TO INPUT THE RESPONSES IN THE ORDER CORRESPONDING TO THE DESIGN MATRIX SETTINGS!

E4. Response Transformations

Regression analysis sometimes indicates the responses should be transformed. PCRSM includes nine commonly used transformations. To perform a transformation on the responses, simply select the option. The original and transformed responses will appear on the screen. The transformed responses do not need to be saved; the regression model will find the transformed data. If you want the responses in original values, select the original values option to delete the transformation.

SECTION 3: REGRESSION

The RUN REGRESSION AND OUTPUT RESULTS option performs the regression calculations and generates tabular and graphical displays of the regression statistics. The output contains aptness and model information. The tables can be sent to a file or printer.

A. APTNESS ASSESSMENT

The aptness assessment option provides graphical information on the standard regression model assumptions (i.e., normality, constant variance).

The first option contains a scatterplot of the standardized residuals versus the predicted value of the response, Y-hat. The second option contains a similar scatterplot containing the residuals versus Y-hat.

Option three plots the ordered standardized residuals versus rankits. The rankits are the expected value of an observation assuming a standard normal distribution. The plot also includes the Wilk-Shapiro test statistic for normality.

B. MODEL RESULTS

PCRSM provides seven tables, including include two ANOVA tables, a coefficient table, the variance-covariance matrix, the correlation matrix, and the design matrix.

B1. ANOVA Table

The Analysis of Variance Table displays the overall and individual sum of squares, F tests, and R^2 statistics. The individual sum of squares are sorted from largest to smallest. Thus, at a quick glance, the variables with the largest explanatory power can be identified.

B2. Coefficient Table

This table displays the coefficient values, standard errors, variances, student t statistics, and P-values. The coefficients are sorted by P-value from most significant variable to least significant.

B3. Variance-Covariance Matrix

The variances of the variables are contained on the diagonal. If the design matrix is orthogonal, the off-diagonal elements will equal zero.

B4. Correlation Matrix

Since each variable is perfectly correlated with itself, the diagonal elements will equal one. If the design is orthogonal, the off-diagonal elements will equal zero.

B5. Lack of Fit ANOVA Table

This table is only available when at least one design point is replicated. It provides the same information as the ANOVA Table, except it replaces the extra sum of squares with lack of fit information. It also includes the lack of fit F statistic and P-value.

B6. Design Matrix

The Design Matrix contains the coded factor settings, the corresponding responses (Y), and the values of the predicted responses (Y-hat).

B7. $(X'X)^{-1}$ Matrix

The $(X'X)^{-1}$ matrix is used in the least squares regression calculations.

Appendix B: Test Case Results

Seven test cases were analyzed to aid the PCRSM verification process. The cases came either from textbooks or were computed on Statistics II to ensure the "truth" was known. This appendix contains the PCRSM output for the seven cases.

Case 1: 2^3 Full Factorial

The first case was a 2^3 full factorial design from Box (4:108-113). It studied the behavior of worsted yarn under cycles of repeated loading.

DESIGN MATRIX

	X0	X1	X2	X3	X12	X13	X23	X123
1	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	-1.00
2	1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00	1.00
3	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00	1.00
4	1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00
5	1.00	-1.00	-1.00	1.00	1.00	-1.00	-1.00	1.00
6	1.00	1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00
7	1.00	-1.00	1.00	1.00	-1.00	-1.00	1.00	-1.00
8	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

	Y	Y-HAT
1	2.83	2.83
2	3.56	3.56
3	2.23	2.23
4	3.06	3.06
5	2.47	2.47

6	3.30	3.30
7	1.95	1.95
8	2.56	2.56

ANALYSIS OF VARIANCE

WARNING ! STATISTICAL TESTS ARE INVALID. SSE = 0 WITH df = 0. THEY HAVE BEEN ARBITRARILY SET TO ONE.

SOURCE	SS	df	MS	F
Regression	2.086	7	0.298	0.30
X1	1.125	1	1.125	1.12
X2	0.696	1	0.696	0.70
X3	0.245	1	0.245	0.24
X13	0.013	1	0.013	0.01
X12	0.003	1	0.003	0.00
X123	0.002	1	0.002	0.00
X23	0.002	1	0.002	0.00
Error	1.000	1	1.000	
Total	2.086	7		

ORTHOGONAL DESIGN

MODEL F VALUE = 0.298 P-VALUE = 0.8904
R SQUARED = 1.000
ADJUSTED R SQUARED =-2.356

COEFFICIENT TABLE

WARNING ! STATISTICAL TESTS ARE INVALID. SSE = 0 WITH df = 0. THEY HAVE BEEN ARBITRARILY SET TO ONE.

VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE

X0	2.745	0.354	0.125	7.764	0.0817
X1	0.375	0.354	0.125	1.061	0.4819
X2	-0.295	0.354	0.125	-0.834	0.5567
X3	-0.175	0.354	0.125	-0.495	0.7070
X123	-0.040	0.354	0.125	-0.113	0.9282
X23	-0.020	0.354	0.125	-0.057	0.9640
X12	-0.015	0.354	0.125	-0.042	0.9730
X13	-0.015	0.354	0.125	-0.042	0.9730

VARIANCE-COVARIANCE MATRIX

	X0	X1	X2	X3	X12	X13	X23
X0	0.125	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.125	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.125	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.125	0.000	0.000	0.000
X12	0.000	0.000	0.000	0.000	0.125	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.125	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.125
X123	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X123
X0	0.000
X1	0.000
X2	0.000
X3	0.000
X12	0.000
X13	0.000
X23	0.000
X123	0.125

CORRELATION MATRIX

	X0	X1	X2	X3	X12	X13	X23	X123
X0	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
X12	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
X123	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000

INV(X'X) MATRIX

	X0	X1	X2	X3	X12	X13	X23	X123
X0	0.125	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.125	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.125	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.125	0.000	0.000	0.000	0.000
X12	0.000	0.000	0.000	0.000	0.125	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.125	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.125	0.000
X123	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.125

Case 2: 2^{5-1} Resolution V Fractional Factorial

The second test case was a 2^{5-1} Resolution V fractional design from Box (4:184-186). It studied the process used in the manufacturing of a drug.

DESIGN MATRIX

X0	X1	X2	X3	X4	X12	X13	X23

1	1.00	-1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00
2	1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00
3	1.00	-1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00
4	1.00	1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00
5	1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00
6	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00
7	1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00
8	1.00	1.00	1.00	1.00	-1.00	1.00	1.00	1.00
9	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00
10	1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00	1.00
11	1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00	-1.00
12	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00
13	1.00	-1.00	-1.00	1.00	1.00	1.00	-1.00	-1.00
14	1.00	1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00
15	1.00	-1.00	1.00	1.00	1.00	-1.00	-1.00	1.00
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

	X14	X24	X34	X5	Y	Y-HAT
1	1.00	1.00	1.00	1.00	51.80	51.40
2	-1.00	1.00	1.00	-1.00	56.30	57.20
3	1.00	-1.00	1.00	-1.00	56.80	58.58
4	-1.00	-1.00	1.00	1.00	48.30	46.02
5	1.00	1.00	-1.00	-1.00	62.30	63.85
6	-1.00	1.00	-1.00	1.00	49.80	47.75
7	1.00	-1.00	-1.00	1.00	49.00	46.07
8	-1.00	-1.00	-1.00	-1.00	46.00	49.42
9	-1.00	-1.00	-1.00	-1.00	72.60	69.18
10	1.00	-1.00	-1.00	1.00	49.50	52.42
11	-1.00	1.00	-1.00	1.00	56.80	58.85
12	1.00	1.00	-1.00	-1.00	63.10	61.55
13	-1.00	-1.00	1.00	1.00	64.60	66.87
14	1.00	-1.00	1.00	-1.00	67.80	66.03
15	-1.00	1.00	1.00	-1.00	70.30	69.40
16	1.00	1.00	1.00	1.00	49.80	50.20

ANALYSIS OF VARIANCE

SOURCE	SS	df	MS	F
Regression	1035.318	11	94.120	5.20

X12	357.210	1	357.210	19.74
X2	344.103	1	344.103	19.02
X5	179.560	1	179.560	9.92
X4	74.823	1	74.823	4.14
X14	36.603	1	36.603	2.02
X3	17.223	1	17.223	0.95
X13	13.322	1	13.322	0.74
X34	9.000	1	9.000	0.50
X24	1.960	1	1.960	0.11
X1	1.210	1	1.210	0.07
X23	0.302	1	0.302	0.02
Error	72.375	4	18.094	
Total	1107.693	15		

ORTHOGONAL DESIGN

MODEL F VALUE = 5.202 P-VALUE = 0.0616
R SQUARED = 0.935
ADJUSTED R SQUARED = 0.804

COEFFICIENT TABLE

VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	57.175	1.063	1.131	53.765	0.0000
X5	-4.725	1.063	1.131	-4.443	0.0113
X4	4.638	1.063	1.131	4.361	0.0121
X1	-3.350	1.063	1.131	-3.150	0.0346
X2	-2.163	1.063	1.131	-2.034	0.1119
X23	-1.513	1.063	1.131	-1.422	0.2283
X34	1.038	1.063	1.131	0.976	0.3837
X14	-0.912	1.063	1.131	-0.858	0.4385
X13	-0.750	1.063	1.131	-0.705	0.5189
X24	0.350	1.063	1.131	0.329	0.7583
X3	0.275	1.063	1.131	0.259	0.8085
X12	0.137	1.063	1.131	0.129	0.9032

VARIANCE-COVARIANCE MATRIX

	X0	X1	X2	X3	X4	X12	X13
X0	1.131	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	1.131	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	1.131	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	1.131	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000	1.131	0.000	0.000
X12	0.000	0.000	0.000	0.000	0.000	1.131	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	1.131
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X14	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X24	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X34	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X5	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X23	X14	X24	X34	X5
X0	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000	0.000
X12	0.000	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000
X23	1.131	0.000	0.000	0.000	0.000
X14	0.000	1.131	0.000	0.000	0.000
X24	0.000	0.000	1.131	0.000	0.000
X34	0.000	0.000	0.000	1.131	0.000
X5	0.000	0.000	0.000	0.000	1.131

CORRELATION MATRIX

	X0	X1	X2	X3	X4	X12	X13	X23
X0	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000

X3	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
X12	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
X14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X24	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X34	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X14	X24	X34	X5
X0	0.000	0.000	0.000	0.000
X1	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000
X12	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000
X14	1.000	0.000	0.000	0.000
X24	0.000	1.000	0.000	0.000
X34	0.000	0.000	1.000	0.000
X5	0.000	0.000	0.000	1.000

INV(X'X) MATRIX

	X0	X1	X2	X3	X4	X12	X13	X23
X0	0.062	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.062	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.062	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.062	0.000	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000	0.062	0.000	0.000	0.000
X12	0.000	0.000	0.000	0.000	0.000	0.062	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.062	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.062
X14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X24	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X34	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X14	X24	X34	X5
X0	0.000	0.000	0.000	0.000
X1	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000
X12	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000
X14	0.062	0.000	0.000	0.000
X24	0.000	0.062	0.000	0.000
X34	0.000	0.000	0.062	0.000
X5	0.000	0.000	0.000	0.062

Case 3: 30 Factor Plackett-Burman

The third case was a 30 factor Plackett-Burman design. The responses were generated randomly. The intent of the example was to ensure the regression model could handle a large problem.

DESIGN MATRIX

	X0	X1	X2	X3	X4	X5	X6	X7
1	1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00
2	1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
3	1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00
4	1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00
5	1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00
6	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00
7	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00
8	1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00

9	1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00
10	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00
11	1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00
12	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00
13	1.00	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00
14	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00	1.00
15	1.00	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00
16	1.00	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00
17	1.00	-1.00	-1.00	1.00	1.00	1.00	1.00	1.00
18	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00
19	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00
20	1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00
21	1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00
22	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00
23	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00
24	1.00	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00
25	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00
26	1.00	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00
27	1.00	-1.00	1.00	-1.00	1.00	1.00	1.00	-1.00
28	1.00	1.00	-1.00	1.00	-1.00	1.00	1.00	1.00
29	1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00	1.00
30	1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00
31	1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00
32	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

	X8	X9	X10	X11	X12	X13	X14	X15
1	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00
2	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00
3	-1.00	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00
4	1.00	-1.00	1.00	-1.00	1.00	1.00	1.00	-1.00
5	-1.00	1.00	-1.00	1.00	-1.00	1.00	1.00	1.00
6	-1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00	1.00
7	-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00
8	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00
9	1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00
10	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
11	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00
12	1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00
13	-1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00
14	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00
15	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00
16	-1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00
17	-1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00
18	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00
19	1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00
20	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00
21	1.00	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00

22	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00	1.00
23	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00
24	-1.00	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00
25	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00	1.00
26	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00
27	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00
28	-1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00
29	1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00
30	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00
31	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00
32	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

	X16	X17	X18	X19	X20	X21	X22	X23
1	-1.00	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00
2	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00	1.00
3	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00
4	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00
5	-1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00
6	1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00
7	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00
8	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00
9	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00
10	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00
11	-1.00	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00
12	1.00	-1.00	1.00	-1.00	1.00	1.00	1.00	-1.00
13	-1.00	1.00	-1.00	1.00	-1.00	1.00	1.00	1.00
14	-1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00	1.00
15	-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00
16	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00
17	1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00
18	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
19	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00
20	1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00
21	-1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00
22	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00
23	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00
24	-1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00
25	-1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00
26	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00
27	1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00
28	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00
29	1.00	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00
30	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00	1.00
31	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00
32	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

	X24	X25	X26	X27	X28	X29	X30	Y
1	-1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00	4.00
2	-1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00	5.00
3	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00	6.00
4	1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00	7.00
5	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00	2.00
6	1.00	1.00	1.00	1.00	-1.00	-1.00	1.00	3.00
7	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00	8.00
8	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00	1.00
9	-1.00	-1.00	1.00	1.00	1.00	1.00	1.00	9.00
10	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00	2.00
11	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	7.00
12	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00	4.00
13	-1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00	5.00
14	1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00	6.00
15	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00	3.00
16	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00	2.00
17	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00	8.00
18	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00
19	-1.00	1.00	-1.00	1.00	1.00	1.00	-1.00	5.00
20	1.00	-1.00	1.00	-1.00	1.00	1.00	1.00	7.00
21	-1.00	1.00	-1.00	1.00	-1.00	1.00	1.00	6.00
22	-1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00	8.00
23	-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	4.00
24	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00	2.00
25	1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	3.00
26	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	1.00	5.00
27	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	9.00
28	1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	4.00
29	-1.00	1.00	-1.00	-1.00	1.00	-1.00	-1.00	1.00
30	1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00	5.00
31	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00	6.00
32	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	4.00

ANALYSIS OF VARIANCE

SOURCE	SS	df	MS	F
Regression	169.500	30	5.650	1.26
X12	15.125	1	15.125	3.36
X8	8.000	1	8.000	1.78
X14	6.125	1	6.125	1.36

X21	3.125	1	3.125	0.69
X22	2.000	1	2.000	0.44
X4	0.500	1	0.500	0.11
X3	0.500	1	0.500	0.11
X18	0.125	1	0.125	0.03
X23	0.000	1	0.000	0.00
X1	0.000	1	0.000	0.00
Error	4.500	1	4.500	
Total	174.000	31		

MODEL F VALUE = 1.256 P-VALUE = 0.6179
R SQUARED = 0.974
ADJUSTED R SQUARED = 0.198

COEFFICIENT TABLE

VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	4.750	0.375	0.141	12.667	0.0502
X13	0.938	0.375	0.141	2.500	0.2426
X21	0.812	0.375	0.141	2.167	0.2756
X14	-0.750	0.375	0.141	-2.000	0.2956
X8	0.688	0.375	0.141	1.833	0.3183
X18	-0.688	0.375	0.141	-1.833	0.3183
X22	-0.562	0.375	0.141	-1.500	0.3748
X4	-0.500	0.375	0.141	-1.333	0.4102
X12	-0.500	0.375	0.141	-1.333	0.4102
X3	0.438	0.375	0.141	1.167	0.4517
X15	0.438	0.375	0.141	1.167	0.4517
X26	0.438	0.375	0.141	1.167	0.4517
X16	0.375	0.375	0.141	1.000	0.4993
X25	-0.375	0.375	0.141	-1.000	0.4993
X11	-0.375	0.375	0.141	-1.000	0.4993
X2	-0.312	0.375	0.141	-0.833	0.5571
X30	0.312	0.375	0.141	0.833	0.5571
X10	0.250	0.375	0.141	0.667	0.6252
X6	-0.250	0.375	0.141	-0.667	0.6252
X28	-0.188	0.375	0.141	-0.500	0.7044
X20	0.125	0.375	0.141	0.333	0.7949
X24	-0.125	0.375	0.141	-0.333	0.7949
X9	0.125	0.375	0.141	0.333	0.7949

X7	-0.125	0.375	0.141	-0.333	0.7949
X17	0.062	0.375	0.141	0.167	0.8947
X5	-0.062	0.375	0.141	-0.167	0.8947
X27	0.062	0.375	0.141	0.167	0.8947
X19	0.062	0.375	0.141	0.167	0.8947
X29	0.062	0.375	0.141	0.167	0.8947
X1	0.000	0.375	0.141	0.000	1.0000
X23	0.000	0.375	0.141	0.000	1.0000

Case 4: 3^3 Full Factorial

The fourth case was a 3^3 full factorial design from Box (4:206-214). It was an extension of Case 1.

DESIGN MATRIX
NOTE: QUADRATICS TERMS HAVE BEEN CORRECTED
SO THEY ARE ORTHOGONAL TO THE MEAN

	X0	X1	X2	X3	X11	X22	X33	X12
1	1.00	-1.00	-1.00	-1.00	0.33	0.33	0.33	1.00
2	1.00	0.00	-1.00	-1.00	-0.67	0.33	0.33	0.00
3	1.00	1.00	-1.00	-1.00	0.33	0.33	0.33	-1.00
4	1.00	-1.00	0.00	-1.00	0.33	-0.67	0.33	0.00
5	1.00	0.00	0.00	-1.00	-0.67	-0.67	0.33	0.00
6	1.00	1.00	0.00	-1.00	0.33	-0.67	0.33	0.00
7	1.00	-1.00	1.00	-1.00	0.33	0.33	0.33	-1.00
8	1.00	0.00	1.00	-1.00	-0.67	0.33	0.33	0.00
9	1.00	1.00	1.00	-1.00	0.33	0.33	0.33	1.00
10	1.00	-1.00	-1.00	0.00	0.33	0.33	-0.67	1.00
11	1.00	0.00	-1.00	0.00	-0.67	0.33	-0.67	0.00
12	1.00	1.00	-1.00	0.00	0.33	0.33	-0.67	-1.00
13	1.00	-1.00	0.00	0.00	0.33	-0.67	-0.67	0.00
14	1.00	0.00	0.00	0.00	-0.67	-0.67	-0.67	0.00
15	1.00	1.00	0.00	0.00	0.33	-0.67	-0.67	0.00
16	1.00	-1.00	1.00	0.00	0.33	0.33	-0.67	-1.00
17	1.00	0.00	1.00	0.00	-0.67	0.33	-0.67	0.00
18	1.00	1.00	1.00	0.00	0.33	0.33	-0.67	1.00
19	1.00	-1.00	-1.00	1.00	0.33	0.33	0.33	1.00
20	1.00	0.00	-1.00	1.00	-0.67	0.33	0.33	0.00

21	1.00	1.00	-1.00	1.00	0.33	0.33	0.33	-1.00
22	1.00	-1.00	0.00	1.00	0.33	-0.67	0.33	0.00
23	1.00	0.00	0.00	1.00	-0.67	-0.67	0.33	0.00
24	1.00	1.00	0.00	1.00	0.33	-0.67	0.33	0.00
25	1.00	-1.00	1.00	1.00	0.33	0.33	0.33	-1.00
26	1.00	0.00	1.00	1.00	-0.67	0.33	0.33	0.00
27	1.00	1.00	1.00	1.00	0.33	0.33	0.33	1.00

	X13	X23	Y	Y-HAT
1	1.00	1.00	2.83	2.76
2	0.00	1.00	3.15	3.21
3	-1.00	1.00	3.56	3.58
4	1.00	0.00	2.53	2.50
5	0.00	0.00	3.01	2.93
6	-1.00	0.00	3.20	3.28
7	1.00	-1.00	2.23	2.26
8	0.00	-1.00	2.65	2.68
9	-1.00	-1.00	3.06	3.01
10	0.00	0.00	2.57	2.66
11	0.00	0.00	3.08	3.07
12	0.00	0.00	3.50	3.41
13	0.00	0.00	2.42	2.39
14	0.00	0.00	2.79	2.79
15	0.00	0.00	3.03	3.11
16	0.00	0.00	2.07	2.14
17	0.00	0.00	2.52	2.52
18	0.00	0.00	2.95	2.83
19	-1.00	-1.00	2.47	2.50
20	0.00	-1.00	2.80	2.88
21	1.00	-1.00	3.30	3.19
22	-1.00	0.00	2.32	2.22
23	0.00	0.00	2.64	2.59
24	1.00	0.00	2.75	2.88
25	-1.00	1.00	1.95	1.96
26	0.00	1.00	2.34	2.32
27	1.00	1.00	2.56	2.59

ANALYSIS OF VARIANCE

SOURCE	SS	df	MS	F
Regression	4.256	9	0.473	66.73

X1	2.352	1	2.352	331.94
X2	1.352	1	1.352	190.75
X3	0.523	1	0.523	73.80
X23	0.011	1	0.011	1.49
X13	0.008	1	0.008	1.17
X33	0.005	1	0.005	0.72
X12	0.003	1	0.003	0.47
X22	0.000	1	0.000	0.14
X11	0.000	1	0.000	0.10
Error	0.120	17	0.007	
Total	4.377	26		

MODEL F VALUE = 66.733 P-VALUE = 0.0000
R SQUARED = 0.972
ADJUSTED R SQUARED = 0.972

COEFFICIENT TABLE

VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	2.751	0.016	0.000	169.816	0.0000
X1	0.361	0.020	0.000	18.219	0.0000
X2	-0.274	0.020	0.000	-13.811	0.0000
X3	-0.170	0.020	0.000	-8.591	0.0000
X13	-0.030	0.024	0.000	-1.223	0.2409
X11	-0.037	0.034	0.001	-1.083	0.2965
X33	-0.029	0.034	0.001	-0.852	0.4080
X12	-0.017	0.024	0.000	-0.683	0.5054
X23	-0.009	0.024	0.000	-0.372	0.7153
X22	0.011	0.034	0.001	0.306	0.7641

VARIANCE-COVARIANCE MATRIX

	X0	X1	X2	X3	X11	X22	X33

X0	0.000	0.000	-0.000	-0.000	0.000	0.000	0.000
X1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X2	-0.000	0.000	0.000	0.000	-0.000	-0.000	-0.000
X3	-0.000	0.000	0.000	0.000	-0.000	-0.000	-0.000
X11	0.000	0.000	-0.000	-0.000	0.001	0.000	0.000
X22	0.000	0.000	-0.000	-0.000	0.000	0.001	0.000
X33	0.000	0.000	-0.000	-0.000	0.000	0.000	0.001
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	-0.000	-0.000	0.000	0.000	0.000

	X12	X13	X23
X0	0.000	0.000	0.000
X1	0.000	0.000	0.000
X2	0.000	0.000	-0.000
X3	0.000	0.000	-0.000
X11	0.000	0.000	0.000
X22	0.000	0.000	0.000
X33	0.000	0.000	0.000
X12	0.000	0.000	0.000
X13	0.000	0.000	0.000
X23	0.000	0.000	0.000

INV(X'X) MATRIX

	X0	X1	X2	X3	X11	X22	X33	X12
X0	0.037	0.000	-0.000	-0.000	0.000	0.000	0.000	0.000
X1	0.000	0.056	0.000	0.000	0.000	0.000	0.000	0.000
X2	-0.000	0.000	0.056	0.000	-0.000	-0.000	-0.000	0.000
X3	-0.000	0.000	0.000	0.056	-0.000	-0.000	-0.000	0.000
X11	0.000	0.000	-0.000	-0.000	0.167	0.000	0.000	0.000
X22	0.000	0.000	-0.000	-0.000	0.000	0.167	0.000	0.000
X33	0.000	0.000	-0.000	-0.000	0.000	0.000	0.167	0.000
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.083
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	-0.000	-0.000	0.000	0.000	0.000	0.000

	X13	X23

X0	0.000	0.000
X1	0.000	0.000
X2	0.000	-0.000
X3	0.000	-0.000
X11	0.000	0.000
X22	0.000	0.000
X33	0.000	0.000
X12	0.000	0.000
X13	0.083	0.000
X23	0.000	0.083

Case 5: 3 Factor Central Composite Design

Case five was a 3 factor central composite design (4:307-312). The experiment explored the conditions leading to maximal elasticity for a certain polymer.

DESIGN MATRIX
NOTE: QUADRATICS TERMS HAVE BEEN CORRECTED
SO THEY ARE ORTHOGONAL TO THE MEAN

	X0	X1	X2	X3	X11	X22	X33	X12
1	1.00	-1.00	-1.00	-1.00	0.00	0.00	0.00	1.00
2	1.00	1.00	-1.00	-1.00	0.00	0.00	0.00	-1.00
3	1.00	-1.00	1.00	-1.00	0.00	0.00	0.00	-1.00
4	1.00	1.00	1.00	-1.00	0.00	0.00	0.00	1.00
5	1.00	-1.00	-1.00	1.00	0.00	0.00	0.00	1.00
6	1.00	1.00	-1.00	1.00	0.00	0.00	0.00	-1.00
7	1.00	-1.00	1.00	1.00	0.00	0.00	0.00	-1.00
8	1.00	1.00	1.00	1.00	0.00	0.00	0.00	1.00
9	1.00	0.00	0.00	0.00	-1.00	-1.00	-1.00	0.00
10	1.00	0.00	0.00	0.00	-1.00	-1.00	-1.00	0.00
11	1.00	-2.00	0.00	0.00	3.00	-1.00	-1.00	0.00
12	1.00	2.00	0.00	0.00	3.00	-1.00	-1.00	0.00
13	1.00	0.00	-2.00	0.00	-1.00	3.00	-1.00	0.00
14	1.00	0.00	2.00	0.00	-1.00	3.00	-1.00	0.00
15	1.00	0.00	0.00	-2.00	-1.00	-1.00	3.00	0.00
16	1.00	0.00	0.00	2.00	-1.00	-1.00	3.00	0.00

	X13	X23	Y	Y-HAT
1	1.00	1.00	25.74	26.83
2	-1.00	1.00	48.98	50.63
3	1.00	-1.00	42.78	42.29
4	-1.00	-1.00	35.94	37.57
5	-1.00	-1.00	41.50	42.45
6	1.00	-1.00	50.10	53.17
7	-1.00	1.00	46.06	46.99
8	1.00	1.00	27.70	29.19
9	0.00	0.00	57.52	57.31
10	0.00	0.00	59.68	57.31
11	0.00	0.00	35.50	35.55
12	0.00	0.00	44.18	41.55
13	0.00	0.00	38.58	36.49
14	0.00	0.00	28.46	27.97
15	0.00	0.00	33.50	32.85
16	0.00	0.00	42.02	40.09

ANALYSIS OF VARIANCE

SOURCE	SS	df	MS	F
Regression	1451.617	9	161.291	23.29
X2	434.307	1	434.307	62.71
X23	406.695	1	406.695	58.73
X13	286.553	1	286.553	41.38
X3	85.543	1	85.543	12.35
X33	72.590	1	72.590	10.48
X1	59.623	1	59.623	8.61
X12	52.418	1	52.418	7.57
X22	36.000	1	36.000	5.20
X11	17.887	1	17.887	2.58
Error	41.551	6	6.925	
Total	1493.168	15		

MODEL F VALUE = 23.291 P-VALUE = 0.0005
R SQUARED = 0.972
ADJUSTED R SQUARED = 0.944

COEFFICIENT TABLE					
VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	41.140	0.658	0.433	62.533	0.0000
X22	-6.270	0.658	0.433	-9.530	0.0000
X33	-5.210	0.658	0.433	-7.919	0.0002
X12	-7.130	0.930	0.866	-7.663	0.0003
X11	-4.690	0.658	0.433	-7.129	0.0004
X13	-3.270	0.930	0.866	-3.515	0.0126
X2	-2.130	0.658	0.433	-3.238	0.0177
X23	-2.730	0.930	0.866	-2.934	0.0261
X3	1.810	0.658	0.433	2.751	0.0332
X1	1.500	0.658	0.433	2.280	0.0628

VARIANCE-COVARIANCE MATRIX

	X0	X1	X2	X3	X11	X22	X33
X0	0.433	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.433	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.433	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.433	0.000	0.000	0.000
X11	0.000	0.000	0.000	0.000	0.433	0.216	0.216
X22	0.000	0.000	0.000	0.000	0.216	0.433	0.216
X33	0.000	0.000	0.000	0.000	0.216	0.216	0.433
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X12	X13	X23
X0	0.000	0.000	0.000
X1	0.000	0.000	0.000
X2	0.000	0.000	0.000
X3	0.000	0.000	0.000
X11	0.000	0.000	0.000

X22	0.000	0.000	0.000
X33	0.000	0.000	0.000
X12	0.866	0.000	0.000
X13	0.000	0.866	0.000
X23	0.000	0.000	0.866

CORRELATION MATRIX

	X0	X1	X2	X3	X11	X22	X33	X12
X0	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
X11	0.000	0.000	0.000	0.000	1.000	0.500	0.500	0.000
X22	0.000	0.000	0.000	0.000	0.500	1.000	0.500	0.000
X33	0.000	0.000	0.000	0.000	0.500	0.500	1.000	0.000
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X13	X23
X0	0.000	0.000
X1	0.000	0.000
X2	0.000	0.000
X3	0.000	0.000
X11	0.000	0.000
X22	0.000	0.000
X33	0.000	0.000
X12	0.000	0.000
X13	1.000	0.000
X23	0.000	1.000

ANALYSIS OF VARIANCE WITH LACK OF FIT

SOURCE	SS	df	MS	F
Regression	1451.617	9	161.291	23.29

Error	41.551	6	6.925	
Lack of Fit	39.218	5	7.844	3.36
Pure Error	2.333	1	2.333	
Total	1493.168	15		

MODEL F VALUE = 23.291 P-VALUE = 0.0005
LACK OF FIT F VALUE = 3.362 P-VALUE = 0.3914
R SQUARED = 0.972
ADJUSTED R SQUARED = 0.944

INV(X'X) MATRIX

	X0	X1	X2	X3	X11	X22	X33	X12
X0	0.062	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.062	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.062	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.062	0.000	0.000	0.000	0.000
X11	0.000	0.000	0.000	0.000	0.062	0.031	0.031	0.000
X22	0.000	0.000	0.000	0.000	0.031	0.062	0.031	0.000
X33	0.000	0.000	0.000	0.000	0.031	0.031	0.062	0.000
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.125
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X13	X23
X0	0.000	0.000
X1	0.000	0.000
X2	0.000	0.000
X3	0.000	0.000
X11	0.000	0.000
X22	0.000	0.000
X33	0.000	0.000
X12	0.000	0.000
X13	0.125	0.000
X23	0.000	0.125

Case 6: 3 Factor CCD with 6 Center Points

The sixth case was a 3 factor central composite design with 6 replicated center points (18:78-82). It investigated the seal strength of a breadwrapper stock.

DESIGN MATRIX
NOTE: QUADRATICS TERMS HAVE BEEN CORRECTED
SO THEY ARE ORTHOGONAL TO THE MEAN

	X0	X1	X2	X3	X11	X22	X33	X12
1	1.00	-1.00	-1.00	-1.00	0.32	0.32	0.32	1.00
2	1.00	1.00	-1.00	-1.00	0.32	0.32	0.32	-1.00
3	1.00	-1.00	1.00	-1.00	0.32	0.32	0.32	-1.00
4	1.00	1.00	1.00	-1.00	0.32	0.32	0.32	1.00
5	1.00	-1.00	-1.00	1.00	0.32	0.32	0.32	1.00
6	1.00	1.00	-1.00	1.00	0.32	0.32	0.32	-1.00
7	1.00	-1.00	1.00	1.00	0.32	0.32	0.32	-1.00
8	1.00	1.00	1.00	1.00	0.32	0.32	0.32	1.00
9	1.00	0.00	0.00	0.00	-0.68	-0.68	-0.68	0.00
10	1.00	0.00	0.00	0.00	-0.68	-0.68	-0.68	0.00
11	1.00	0.00	0.00	0.00	-0.68	-0.68	-0.68	0.00
12	1.00	0.00	0.00	0.00	-0.68	-0.68	-0.68	0.00
13	1.00	0.00	0.00	0.00	-0.68	-0.68	-0.68	0.00
14	1.00	0.00	0.00	0.00	-0.68	-0.68	-0.68	0.00
15	1.00	-1.68	0.00	0.00	2.15	-0.68	-0.68	0.00
16	1.00	1.68	0.00	0.00	2.15	-0.68	-0.68	0.00
17	1.00	0.00	-1.68	0.00	-0.68	2.15	-0.68	0.00
18	1.00	0.00	1.68	0.00	-0.68	2.15	-0.68	0.00
19	1.00	0.00	0.00	-1.68	-0.68	-0.68	2.15	0.00
20	1.00	0.00	0.00	1.68	-0.68	-0.68	2.15	0.00

	X13	X23	Y	Y-HAT
1	1.00	1.00	6.60	6.51
2	-1.00	1.00	6.90	6.00
3	1.00	-1.00	7.90	7.08
4	-1.00	-1.00	6.10	5.18
5	-1.00	-1.00	9.20	9.25

6	1.00	-1.00	6.80	6.74
7	-1.00	1.00	10.40	10.43
8	1.00	1.00	7.30	6.52
9	0.00	0.00	10.10	10.16
10	0.00	0.00	9.90	10.16
11	0.00	0.00	12.20	10.16
12	0.00	0.00	9.70	10.16
13	0.00	0.00	9.70	10.16
14	0.00	0.00	9.60	10.16
15	0.00	0.00	9.80	9.87
16	0.00	0.00	5.00	6.16
17	0.00	0.00	6.90	7.07
18	0.00	0.00	6.30	7.36
19	0.00	0.00	4.00	5.20
20	0.00	0.00	8.60	8.63

ANALYSIS OF VARIANCE

SOURCE	SS	df	MS	F
Regression	70.311	9	7.812	6.59
X13	19.014	1	19.014	16.03
X33	16.636	1	16.636	14.03
X22	14.222	1	14.222	11.99
X11	12.546	1	12.546	10.58
X3	4.629	1	4.629	3.90
X23	2.000	1	2.000	1.69
X12	0.980	1	0.980	0.83
X2	0.180	1	0.180	0.15
X1	0.104	1	0.104	0.09
Error	11.859	10	1.186	
Total	82.170	19		

MODEL F VALUE = 6.587   P-VALUE = 0.0033
R SQUARED = 0.856
ADJUSTED R SQUARED = 0.856

COEFFICIENT TABLE

VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	8.150	0.244	0.059	33.469	0.0000
X33	-1.148	0.287	0.082	-4.004	0.0025
X1	-1.104	0.295	0.087	-3.745	0.0038
X22	-1.042	0.287	0.082	-3.634	0.0046
X3	1.020	0.295	0.087	3.463	0.0061
X11	-0.760	0.287	0.082	-2.649	0.0243
X13	-0.500	0.385	0.148	-1.299	0.2246
X12	-0.350	0.385	0.148	-0.909	0.3858
X23	0.150	0.385	0.148	0.390	0.7055
X2	0.087	0.295	0.087	0.296	0.7738

VARIANCE-COVARIANCE MATRIX

	X0	X1	X2	X3	X11	X22	X33
X0	0.059	0.000	0.000	0.000	-0.000	-0.000	-0.000
X1	0.000	0.087	-0.000	-0.000	0.000	-0.000	-0.000
X2	0.000	-0.000	0.087	-0.000	-0.000	0.000	-0.000
X3	0.000	-0.000	-0.000	0.087	-0.000	-0.000	0.000
X11	-0.000	0.000	-0.000	-0.000	0.082	0.008	0.008
X22	-0.000	-0.000	0.000	-0.000	0.008	0.082	0.008
X33	-0.000	-0.000	-0.000	0.000	0.008	0.008	0.082
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X12	X13	X23
X0	0.000	0.000	0.000
X1	0.000	0.000	0.000
X2	0.000	0.000	0.000
X3	0.000	0.000	0.000
X11	0.000	0.000	0.000
X22	0.000	0.000	0.000
X33	0.000	0.000	0.000
X12	0.148	0.000	0.000
X13	0.000	0.148	0.000

X23	0.000	0.000	0.148

ANALYSIS OF VARIANCE WITH LACK OF FIT

SOURCE	SS	df	MS	F
Regression	70.311	9	7.812	6.59
Error	11.859	10	1.186	
Lack of Fit	6.899	5	1.380	1.39
Pure Error	4.960	5	0.992	
Total	82.170	19		

MODEL F VALUE = 6.587 P-VALUE = 0.0033
LACK OF FIT F VALUE = 1.391 P-VALUE = 0.3603
R SQUARED = 0.856
ADJUSTED R SQUARED = 0.856

INV(X'X) MATRIX

	X0	X1	X2	X3	X11	X22	X33	X12
X0	0.050	0.000	0.000	0.000	-0.000	-0.000	-0.000	0.000
X1	0.000	0.073	-0.000	-0.000	0.000	-0.000	-0.000	0.000
X2	0.000	-0.000	0.073	-0.000	-0.000	0.000	-0.000	0.000
X3	0.000	-0.000	-0.000	0.073	-0.000	-0.000	0.000	0.000
X11	-0.000	0.000	-0.000	-0.000	0.069	0.007	0.007	0.000
X22	-0.000	-0.000	0.000	-0.000	0.007	0.069	0.007	0.000
X33	-0.000	-0.000	-0.000	0.000	0.007	0.007	0.069	0.000
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.125
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X13	X23
X0	0.000	0.000

X1	0.000	0.000
X2	0.000	0.000
X3	0.000	0.000
X11	0.000	0.000
X22	0.000	0.000
X33	0.000	0.000
X12	0.000	0.000
X13	0.125	0.000
X23	0.000	0.125

Case 7: 4 Factor Box-Behnken

The seventh case was a 4 factor Box-Behnken design (4:223).

DESIGN MATRIX
NOTE: QUADRATICS TERMS HAVE BEEN CORRECTED
SO THEY ARE ORTHOGONAL TO THE MEAN

	X0	X1	X2	X3	X4	X11	X22	X33
1	1.00	-1.00	-1.00	0.00	0.00	0.56	0.56	-0.44
2	1.00	1.00	-1.00	0.00	0.00	0.56	0.56	-0.44
3	1.00	-1.00	1.00	0.00	0.00	0.56	0.56	-0.44
4	1.00	1.00	1.00	0.00	0.00	0.56	0.56	-0.44
5	1.00	0.00	0.00	-1.00	-1.00	-0.44	-0.44	0.56
6	1.00	0.00	0.00	1.00	-1.00	-0.44	-0.44	0.56
7	1.00	0.00	0.00	-1.00	1.00	-0.44	-0.44	0.56
8	1.00	0.00	0.00	1.00	1.00	-0.44	-0.44	0.56
9	1.00	-1.00	0.00	0.00	-1.00	0.56	-0.44	-0.44
10	1.00	1.00	0.00	0.00	-1.00	0.56	-0.44	-0.44
11	1.00	-1.00	0.00	0.00	1.00	0.56	-0.44	-0.44
12	1.00	1.00	0.00	0.00	1.00	0.56	-0.44	-0.44
13	1.00	0.00	-1.00	-1.00	0.00	-0.44	0.56	0.56
14	1.00	0.00	1.00	-1.00	0.00	-0.44	0.56	0.56
15	1.00	0.00	-1.00	1.00	0.00	-0.44	0.56	0.56
16	1.00	0.00	1.00	1.00	0.00	-0.44	0.56	0.56
17	1.00	-1.00	0.00	-1.00	0.00	0.56	-0.44	0.56
18	1.00	1.00	0.00	-1.00	0.00	0.56	-0.44	0.56
19	1.00	-1.00	0.00	1.00	0.00	0.56	-0.44	0.56
20	1.00	1.00	0.00	1.00	0.00	0.56	-0.44	0.56
21	1.00	0.00	-1.00	0.00	-1.00	-0.44	0.56	-0.44

22	1.00	0.00	1.00	0.00	-1.00	-0.44	0.56	-0.44
23	1.00	0.00	-1.00	0.00	1.00	-0.44	0.56	-0.44
24	1.00	0.00	1.00	0.00	1.00	-0.44	0.56	-0.44
25	1.00	0.00	0.00	0.00	0.00	-0.44	-0.44	-0.44
26	1.00	0.00	0.00	0.00	0.00	-0.44	-0.44	-0.44
27	1.00	0.00	0.00	0.00	0.00	-0.44	-0.44	-0.44

	X44	X12	X13	X23	X14	X24	X34	Y
1	-0.44	1.00	0.00	0.00	0.00	0.00	0.00	84.70
2	-0.44	-1.00	0.00	0.00	0.00	0.00	0.00	93.30
3	-0.44	-1.00	0.00	0.00	0.00	0.00	0.00	84.20
4	-0.44	1.00	0.00	0.00	0.00	0.00	0.00	86.10
5	0.56	0.00	0.00	0.00	0.00	0.00	1.00	85.70
6	0.56	0.00	0.00	0.00	0.00	0.00	-1.00	96.40
7	0.56	0.00	0.00	0.00	0.00	0.00	-1.00	88.10
8	0.56	0.00	0.00	0.00	0.00	0.00	1.00	81.80
9	0.56	0.00	0.00	0.00	1.00	0.00	0.00	89.40
10	0.56	0.00	0.00	0.00	-1.00	0.00	0.00	88.70
11	0.56	0.00	0.00	0.00	-1.00	0.00	0.00	77.80
12	0.56	0.00	0.00	0.00	1.00	0.00	0.00	80.90
13	-0.44	0.00	0.00	1.00	0.00	0.00	0.00	80.90
14	-0.44	0.00	0.00	-1.00	0.00	0.00	0.00	79.80
15	-0.44	0.00	0.00	-1.00	0.00	0.00	0.00	86.80
16	-0.44	0.00	0.00	1.00	0.00	0.00	0.00	79.00
17	-0.44	0.00	1.00	0.00	0.00	0.00	0.00	79.70
18	-0.44	0.00	-1.00	0.00	0.00	0.00	0.00	92.50
19	-0.44	0.00	-1.00	0.00	0.00	0.00	0.00	89.40
20	-0.44	0.00	1.00	0.00	0.00	0.00	0.00	86.90
21	0.56	0.00	0.00	0.00	0.00	1.00	0.00	86.10
22	0.56	0.00	0.00	0.00	0.00	-1.00	0.00	87.90
23	0.56	0.00	0.00	0.00	0.00	-1.00	0.00	85.10
24	0.56	0.00	0.00	0.00	0.00	1.00	0.00	76.40
25	-0.44	0.00	0.00	0.00	0.00	0.00	0.00	93.80
26	-0.44	0.00	0.00	0.00	0.00	0.00	0.00	87.30
27	-0.44	0.00	0.00	0.00	0.00	0.00	0.00	90.70

ANALYSIS OF VARIANCE

SOURCE	SS	df	MS	F
Regression	563.221	14	40.230	3.81
X13	162.068	1	162.068	15.35

X14	72.250	1	72.250	6.84
X1	62.438	1	62.438	5.91
X3	58.523	1	58.523	5.54
X22	46.021	1	46.021	4.36
X4	44.853	1	44.853	4.25
X24	35.422	1	35.422	3.35
X11	27.562	1	27.562	2.61
X34	15.413	1	15.413	1.46
X44	11.500	1	11.500	1.09
Error	126.734	12	10.561	
Total	689.956	26		

MODEL F VALUE = 3.809 P-VALUE = 0.0122
R SQUARED = 0.816
ADJUSTED R SQUARED = 0.633

COEFFICIENT TABLE

VARIABLE	VALUE	STD ERROR	VARIANCE	STUDENT-T	P-VALUE
X0	85.904	0.625	0.391	137.353	0.0000
X4	-3.675	0.938	0.880	-3.917	0.0020
X22	-4.329	1.407	1.980	-3.076	0.0096
X34	-4.250	1.625	2.640	-2.616	0.0225
X13	-3.825	1.625	2.640	-2.354	0.0364
X2	-1.958	0.938	0.880	-2.087	0.0587
X1	1.933	0.938	0.880	2.061	0.0615
X44	-2.579	1.407	1.980	-1.833	0.0915
X24	-2.625	1.625	2.640	-1.615	0.1318
X33	-2.242	1.407	1.980	-1.593	0.1393
X3	1.133	0.938	0.880	1.208	0.2522
X23	-1.675	1.625	2.640	-1.031	0.3246
X12	-1.675	1.625	2.640	-1.031	0.3246
X11	-1.417	1.407	1.980	-1.007	0.3356
X14	0.950	1.625	2.640	0.585	0.5707

VARIANCE-COVARIANCE MATRIX

	X0	X1	X2	X3	X4	X11	X22
X0	0.391	0.000	0.000	0.000	0.000	-0.000	-0.000
X1	0.000	0.880	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.880	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.880	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000	0.880	0.000	0.000
X11	-0.000	0.000	0.000	0.000	0.000	1.980	0.660
X22	-0.000	0.000	0.000	0.000	0.000	0.660	1.980
X33	-0.000	0.000	0.000	0.000	0.000	0.660	0.660
X44	-0.000	0.000	0.000	0.000	0.000	0.660	0.660
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X14	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X24	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X34	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X33	X44	X12	X13	X23	X14	X24
X0	-0.000	-0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X11	0.660	0.660	0.000	0.000	0.000	0.000	0.000
X22	0.660	0.660	0.000	0.000	0.000	0.000	0.000
X33	1.980	0.660	0.000	0.000	0.000	0.000	0.000
X44	0.660	1.980	0.000	0.000	0.000	0.000	0.000
X12	0.000	0.000	2.640	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	2.640	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	2.640	0.000	0.000
X14	0.000	0.000	0.000	0.000	0.000	2.640	0.000
X24	0.000	0.000	0.000	0.000	0.000	0.000	2.640
X34	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X34
X0	0.000
X1	0.000
X2	0.000
X3	0.000
X4	0.000

X11	0.000
X22	0.000
X33	0.000
X44	0.000
X12	0.000
X13	0.000
X23	0.000
X14	0.000
X24	0.000
X34	2.640

ANALYSIS OF VARIANCE WITH LACK OF FIT

SOURCE	SS	df	MS	F
Regression	563.221	14	40.230	3.81
Error	126.734	12	10.561	
Lack of Fit	105.594	10	10.559	1.00
Pure Error	21.140	2	10.570	
Total	689.956	26		

MODEL F VALUE = 3.809 P-VALUE = 0.0122
LACK OF FIT F VALUE = 0.999 P-VALUE = 0.5953
R SQUARED = 0.816
ADJUSTED R SQUARED = 0.633

INV(X'X) MATRIX

	X0	X1	X2	X3	X4	X11	X22	X33
X0	0.037	0.000	0.000	0.000	0.000	-0.000	-0.000	-0.000
X1	0.000	0.083	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.083	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.083	0.000	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000	0.083	0.000	0.000	0.000
X11	-0.000	0.000	0.000	0.000	0.000	0.188	0.063	0.063
X22	-0.000	0.000	0.000	0.000	0.000	0.063	0.188	0.063

X33	-0.000	0.000	0.000	0.000	0.000	0.063	0.063	0.188
X44	-0.000	0.000	0.000	0.000	0.000	0.063	0.063	0.063
X12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X24	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X34	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

	X44	X12	X13	X23	X14	X24	X34
X0	-0.000	0.000	0.000	0.000	0.000	0.000	0.000
X1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X2	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X3	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
X11	0.063	0.000	0.000	0.000	0.000	0.000	0.000
X22	0.063	0.000	0.000	0.000	0.000	0.000	0.000
X33	0.063	0.000	0.000	0.000	0.000	0.000	0.000
X44	0.188	0.000	0.000	0.000	0.000	0.000	0.000
X12	0.000	0.250	0.000	0.000	0.000	0.000	0.000
X13	0.000	0.000	0.250	0.000	0.000	0.000	0.000
X23	0.000	0.000	0.000	0.250	0.000	0.000	0.000
X14	0.000	0.000	0.000	0.000	0.250	0.000	0.000
X24	0.000	0.000	0.000	0.000	0.000	0.250	0.000
X34	0.000	0.000	0.000	0.000	0.000	0.000	0.250

Appendix C: Two Level Designs

The computer program LEVEL2.C generates two level designs based on the number of design factors, resolution, number of replications, and number of center points. It is called by the DESIGN CHOICES menu in PCRSM.

Three two level design types are available, including full factorials, fractional factorials, and Plackett-Burman designs. The fractional factorial designs were developed based on a table generated by Box, Hunter, and Hunter (4:164-165). The table provides useful aliasing structures to generate designs for 3 to 11 factors, with resolutions ranging from III to VIII and between 4 and 128 runs. Although the regression package can only handle 32 run designs, the LEVEL2.C program generates up to 64 run designs.

Plackett and Burman provide orthogonal or nearly orthogonal designs for up to 100 factors (2:5-15). This program generates designs for up to 39 factors. The Plackett-Burman designs are resolution III.

For each design, a matrix is initially created that includes only the main factors at each of their settings.

The interaction terms between the factors are generated using the Bauer Binary Count Technique (2:4-13). First, the interactions between the first two factors are generated,

then the interactions between the first two factors and the third factor are generated, and so on. When replications are requested, duplicates of the design matrix are copied to create the extra runs.

When center points are requested, the appropriate number of rows of zeros are placed after the design matrix with or without repetitions.

Appendix D: Three Level Designs

The program LEVEL3.C creates three level designs based on the number of factors, design choice, number of replications, and number of center points. It is called by the DESIGN CHOICES menu in PCRSM.

The three available design choices are full factorials, central composite designs (CCD), and Box-Behnken designs. The designs are available for 2-6 factors.

The CCD designs expand a two level design by adding axial and center points to create a three level design. When the axial points, or alpha levels, are properly placed, orthogonality can be maintained. Unless otherwise directed, the program uses the orthogonal settings.

The Box-Behnken designs are nearly orthogonal three level designs that require less than the full factorial number of runs to get second order information (2:7-20).

For each design, a matrix is initially created that includes only the main factors at each of their settings.

The interaction terms between the factors are generated using the Bauer Binary Count Technique (2:4-13). First, the interactions between the first two factors are generated, then the interactions between the first two factors and the third factor are generated, and so on.

When replications are requested for full factorial and Box-Behnken designs, duplicates of the design matrix are repeated immediately after the initial design matrix to create the extra runs.

When center points are requested for full factorial or Box-Behnken designs, the appropriate number of rows of zeros are placed after the design matrix with or without repetitions.

When center points are requested for CCD designs, the center points are added immediately after the center point in the design. When replications are asked for, the entire design, including the extra center points, is repeated.

Appendix E: Regression Model

This appendix discusses the regression model in the decision support system. PCRSM uses linear regression to conduct group and factor screening as well as to identify the appropriate meta-model. Three computer programs were written in C to support the RSM analysis. The first program REGDAT.C loads the experimental design matrix and responses. It allows the analyst to choose the variables to regress and perform transformations on the responses. The second program, REGRESS.C, performs the least squares regression. The third program, REGOUT.C generates results and output.

The flow diagrams in Figures 2 and 3 depict the processes of the regression phase of the analysis. This chapter discusses each of the processes.

Data Input

The data management program REGDAT.C reads in the design matrix (X) and response (Y) information and passes it to the regression program REGRESS.C. REGDAT.C is called by the REGRESSION INPUTS menu in PCRSM.

REGDAT.C. Figure 2 shows the activities that occur in the program REGDAT.C. The four options are to input the design matrix created in the experimental design phase, select some or all of the variables for regression, input

the response vector, perform transformations on the response vector, and to exit the program.

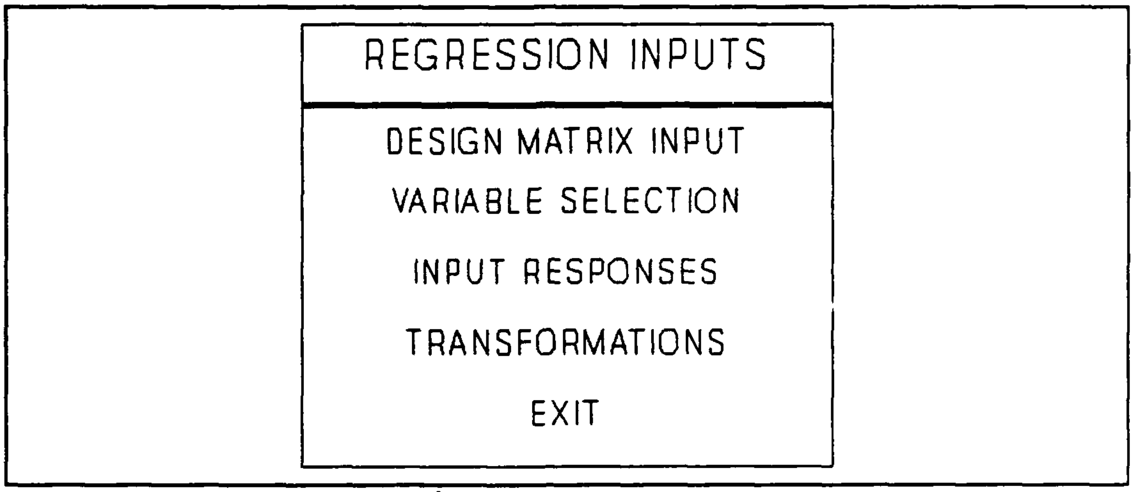


Figure 2. REGDAT.C

Design Matrix Input. Three choices are available from this screen. A previously saved design or the latest design generated in experimental design can be chosen.

Design Matrix Variable Selection . The design matrix automatically generates all the main terms, quadratic terms (when applicable), and appropriate interactions. However, there are very few times when all the terms are significant. Furthermore, when all the terms are regressed in a two level design without center points or repetitions, there are an equal number of variables and runs. This leads to a perfect fit and invalidates many of the statistical tests because the sum of squares for error (SSE) is zero. The variable selection option allows the analyst to choose the regression terms.

Response Vector Input. The responses, or outputs, requested by the experimental design are input here. The responses can be entered by hand at the keyboard or read from an ASCII file.

The ASCII file must be in a specific format. The first line must contain the number of responses, say ten. The second through eleventh row (the ten responses) each must contain one response per line.

Response Vector Transformation. Sometimes regression analysis indicates that a transformation should be performed on the responses. This screen provides nine options. The responses are called Y. The nine options include

1) power transformations (Y**alpha)
2) natural log of Y
3) log base 10 of Y
4) arcsin of the square root of Y
5) natural log of (1 + Y)/(1 - Y)
6) inverse of Y (i.e., 1/Y)
7) square root of Y
8) square of Y (i.e., Y**2)
9) natural log of (B - Y) where B is some value

REGRESS.C. The results of REGDAT.C are sent directly to the regression program (see Block 1 of Figure 3). This block reads the REGDAT.C information, declares and

initializes the variables used in the program, and conducts some preliminary tests.

The regression program is limited to processing 35 rows, or runs, and 35 columns, or variables. This limitation is a result of the limited RAM memory available in personal computers combined with the large number of matrix calculations required in the least squares calculations. The regression program checks to ensure that no more than 35 row or columns are read and processed.

Degrees of Freedom

This section discusses degrees of freedom for the model and lack of fit (see Block 2 of Figure 3). As discussed in Chapter 2, least squares regression determines the relationship between the design matrix and the responses. Further, it does so by calculating sum of squares statistics. There are three components of the sum of squares. They are the sum of squares of regression (SSR), sum of squares of error (SSE) and total sum of squares (SSTO). The addition of SSR and SSE equals SSTO.

The general linear model chosen for this regression analysis corrects for the mean sum of squares. (19:90) This takes away one degree of freedom from SSR and SSTO. The corrected sum of squares are referred to as SSRc, SSEc, and SSTOc.

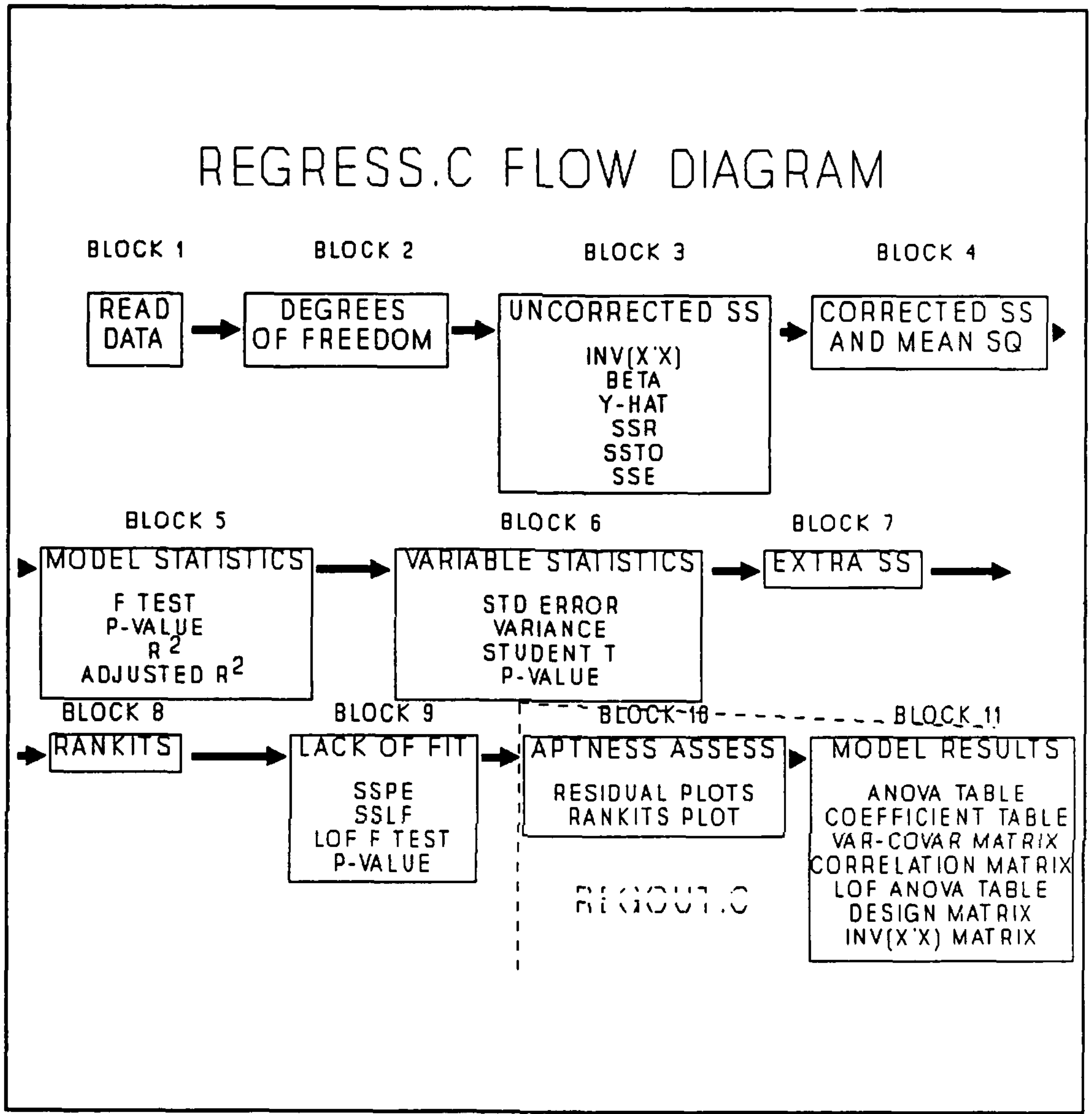


Figure 3. Regression Flow Diagram

Let m be the total number of runs and n be the number of regression terms. The degrees of freedom for SSRc = n - 1. The degrees of freedom for SSEc = m - n. The degrees of freedom for SSTOc = m - 1.

than one run is made at the same design setting), model lack of fit can be tested. Lack of fit breaks the sum of

squares for error (SSE) into two components, sum of squares for pure error and sum of squares for lack of fit. Assuming the number of different design settings equals c, the degrees of freedom associated with lack of fit is c - n. The degrees of freedom associated with pure error is n - c.

Uncorrected Sum of Squares

The uncorrected sum of squares calculations shown in Block 3 of Figure 3, except where noted otherwise, follow the steps and matrix multiplications performed in Sparrow's thesis. (28:66)

$(X'X)^{-1}$. When the design matrix (called X) is non-orthogonal, such as many of the Plackett-Burman designs and three levels designs with center points, X'X must be inverted before the coefficients of variables can be calculated. The algorithm used was published by Conte (6:163-168). The algorithm was written in Fortran and then rewritten in C.

When the design matrix is orthogonal and follows a pattern, the $(X'X)^{-1}$ can be calculated directly. This calculation is used for the orthogonal two level designs and the three level central composite designs (CCD). For orthogonal designs, the entire matrix is zeros except along the diagonal.

For two level orthogonal designs, the upper left hand entry is 1/m. The remaining entries are 1/(m - number of

center points). When there are no center points, they are all the same. The calculations are more complicated for the CCD designs (18:134).

Beta. The coefficients of the variables are calculated by $(X'X)^{-1}X'Y$.

Y-hat. The value of the response predicted by the regression model is called Y-hat. It is calculated by X*Beta.

SSR. The sum of squares for regression (SSR) is calculated by the equation Beta'X'Y.

SSTO. The total sum of squares for the model is calculated by Y'Y.

SSE. The sum of squares for error (SSE) is the difference between SSTO and SSR.

Corrected Sum of Squares and Mean Squares

As previously discussed, the sum of squares are corrected for the mean. The correction factor is mY^2 (28:69). Thus, SSRc is SSR - mY^2. SSTOc = SSTO - mY^2. And SSEc is unchanged from SSE.

The regression mean square (MSRc) and error mean square (MSEc) are calculated by dividing the sum of squares by the associated degrees of freedom. MSRc = SSRc / SSRc d.f., and MSEc = SSEc / SSEc d.f.

Model Statistics

Block 5 of Figure 3 shows several statistics that assess the adequacy of the model. The F test assesses the hypothesis that all the coefficients in the model (except the Bo term) are equal to zero. The test statistics is F = MSRc/MSEc. If F is large, the test is rejected and at least one coefficient is not equal to zero.

A P-value is generated to determine what alpha level is required to pass the F test. The P-value calculations were translated from Numerical Recipes (20:157-169).

The coefficient of multiple determination, R^2, "measures the proportionate reduction of total variation in Y associated with the use of the set of X variables" (19:241). It is the ratio of SSRc to SSTOc. When Y is well explained by the model, SSRc is large compared to SSEc. Thus, the ratio of SSRc to SSTOc is close to one. However, as variables are added to a model, R^2 will increase. To offset the effect of increasing R^2 by adding variables, it is adjusted by the associated degree of freedom for SSEc and SSTOc. The equations for $R^2_a = 1 - (m-1)/(m-n)*SSE/SSTO$.

<u>Variable Statistics</u>

Block 6 of Figure 3 shows the statistics that provide information on the regression variables. The coefficients of the variables are the Beta values previously discussed. The variances of the coefficients are calculated by multiplying the $(X'X)^{-1}$ matrix times MSEc. The standard

errors are the square root of the variances. The Student T test statistic is calculated by dividing the coefficient values by its standard error. The hypothesis for the Student T test is that the value of the coefficient equals zero. The P-value for the T test determines what alpha level is required to reject the test.

Individual Sum of Squares

The sum of squares for regression (SSRc) can be broken down into individual, or extra, sum of squares corresponding to each of the variables in the model. The extra sum of squares reflect the reduction in SSEc by adding another variable, given the variables already in the model (19:282).

When the design matrices are non-orthogonal, the extra sums are order dependent. Since some or all of the variables are dependent, the extra sum of squares calculated for a variable may include extra sum of squares due to another variable as well. However, when designs are orthogonal, the variables are independent and the extra sum of squares calculated for each variable is due solely to the addition of that variable to the model.

In general, the extra sum of squares are calculated by adding variables into the model one at a time, each time calculating the overall SSRc. The extra sum of squares for a particular variable is the difference between the SSRc with that variable in the model and the previous SSRc

without that variable. Because of the computational difficulty and time consumed to recalculate a new SSRc for each extra sum of squares, the program only allows up to ten extra sum of squares calculations for non-orthogonal designs.

A special case occurs when the design is orthogonal. The diagonal elements of the X'*Y*Beta' matrix contain the individual sum of squares corresponding to each of the model variables (2:4-14). Since it is a one step calculation, all the extra sum of squares are provided for orthogonal designs.

Rankits

Rankits (shown in Block 8 of Figure 3) are the expected value of an observation at a given percentile of a distribution. Blom (23:161) proposed that the formula z[(r-.375)/(n+.25)] is a good approximation to the standard normal distribution. N is the sample size, r is the rth ordered statistic, and z is the percentile of the standard normal distribution.

Rankits are used to test the assumption of normality for the model discussed in Chapter 2. If the model follows the normal distribution, the ordered standardized residuals of the model should nearly equal to the corresponding rankits. When the two sets are plotted against each other,

they should lie in a straight line at approximately a forty-five degree angle.

Lack of Fit

When more than one set of data is collected at a given design setting (i.e., replications), lack of fit information can be obtained. Lack of fit testing is used to determine whether or not a given model adequately fits the data. Lack of fit breaks SSEc down into two components, sum of squares for pure error (SSPE) and sum of squares for lack of fit (SSLF). SSPE measures the deviations of each of the replicated points from the average of the replicated points. SSLF measures how far the average of the replicated points is from the regression line. The mean squared error of SSPE and SSLF is obtained by dividing by the associated degrees of freedom discussed at the beginning of this chapter.

The hypothesis is that the model adequately describes the data. The lack of fit F test statistic is MSLF/MSPE (19:130). If the F statistic is large, there is significantly more lack of fit than pure error, and the model does not fit well.
The P-value for the lack of fit F test determines what alpha level is required to reject the hypothesis.

A significant amount of computer code is dedicated to the lack of fit tests. This is caused by the fact that the designs differ in structure. In order to compute SSLF and

SSPE, the replicated design points need to be identified. The different designs have different patterns of center points and replications. Thus, separate code was written for two level designs, central composite designs, and the remaining three level designs.

Output

The REGRESS.C program passes the regression statistics to the program REGOUT.C, where the data is formatted for output to the computer screen, file, and/or printer. This section describes the various outputs.

Aptness Output. The aptness tests shown in Block 10 of Figure 3 are primarily concerned with the assumptions of constancy of variance and normality, as well as the presence of outliers.

The aptness information is presented graphically in the form of a scatterplot. Y-hat, the predicted values of Y, are plotted against the residuals or standardized residuals to make a variance assessment. The ordered standardized residuals are plotted against the rankits to make a normality assessment.

The normality assessment is supported by the Wilk-Shapiro statistic. The Wilk-Shapiro statistic compares the ordered standardized residuals to the rankits and generates a statistic (26:215). The value of the statistic ranges

from zero to one. A value close to one supports the normality assumption.

Model Output. Figure 11 of Figure 3 lists the various regression model outputs.

ANOVA Table. The Analysis of Variance Table displays the overall and extra sum of squares, F tests, and R^2 statistics.

Coefficient Table. The Coefficient Table displays the coefficient values, standard errors, variances, Student T statistics, and P-values.

Variance-Covariance Matrix. The variances lie on the diagonal and the covariances between the variables lie off the diagonal. When the design is orthogonal, there will be no off diagonal elements.

Correlation Matrix. The correlation between any variable and itself is equal to one. Thus, the diagonal elements are equal to one. There ar no off diagonal elements for orthogonal designs.

Lack of Fit ANOVA Table. The Lack of Fit Analysis of Variance Table is only available when at least one design point is repeated. It provides the same information as the ANOVA Table, except it replaces the extra sum of squares with lack of fit information. It also includes the lack of fit F statistic and P-value.

Design Matrix. The Design Matrix lists the coded factor settings, the corresponding responses, and predicted responses.

$(X'X)^{-1}$ Matrix. The $(X'X)^{-1}$ Matrix is an nxn matrix that corresponds to each of the variables in the model. When the design is orthogonal, only diagonal entries exist.

Appendix F: Computer Code

PCRSM is run with five executable codes, PCRSM.EXE, MASTER.EXE, REGRESS.EXE, REGOUT.EXE, AND CLIPRAW.EXE. MASTER is written in Clipper and C. The other programs are written in C. This appendix contains a listing of the computer code for each of the programs.

PCRSM.EXE

```
/*  RSM.C -- DRIVER MENU FOR RSM IN EXPERIMENTAL DESIGN AND
REGRESSION */
/*  THIS PROGRAM NEEDS ANSI.SYS LOADED IN THE CONFIG.SYS FILE */
/*  AS IN "DEVICE=ANSI.SYS" */

# include <string.h>
# include <stdlib.h>
# include <dos.h>
# include <io.h>
# include <stdio.h>
#define TRUE 1
#define NUM 3
#define CLEAR   "\x1B[2J"     /* clears screen */
#define ERASE   "\x1B[K"      /* erases line from cursor position
*/
#define NORMAL  "\x1B[0m"     /* normal attribute */
#define BOLD    "\x1B[1m"
#define BLINK   "\x1B[5m"
#define REVERSE "\x1B[7m"     /* reverse viedo attribute */
#define LEFTCOL "\x1B[%dC"
#define ROWCOL  "\x1B[%d;%df"
#define HOME    "\x1B[11;20f" /* the starting position of the
menu */
#define BOTTOM  "\x1B[24;1f"  /* the starting position of the
message line */
#define U_ARROW 72            /* the extended code for up arrow
*/
#define D_ARROW 80            /* the extended code for down arrow
*/
#define RETURN  13            /* the code for carriage return */
#define BLOCK    "\xDB"
#define UHALF    "\xDF"
#define LHALF    "\xDC"
#define ULCORNER "\xC9"
```

```
#define URCORNER "\xBB"
#define LLCORNER "\xC8"
#define LRCORNER "\xBC"
#define HDOUBLE  "\xCD"
#define VDOUBLE  "\xBA"
int return_val;
char mas_filename[12] = "master.exe";
char reg_filename[12] = "regress.exe";
char out_filename[12] = "regout.exe";
struct find_t exe_file;

main()
{
static char *items[NUM] =
      { "EXPERIMENTAL DESIGN AND REGRESSION INPUTS",
        "    RUN REGRESSION AND OUTPUT RESULTS    ",
        "                 QUIT                    ", };
static char *message[NUM] =
      { "SELECT TO CREATE EXPERIMENTAL DESIGNS AND CREATE
REGRESSION INPUTS",
        "SELECT TO RUN REGRESSION",
        "SELECT TO QUIT TO DOS", };

int curpos;
int code;

/* write initial program description screen */
fscreen();

curpos=0;
while (TRUE) {
  printf(CLEAR);
  display_box();
  display(items,message,NUM,curpos);
  code = getcode();
  switch (code)
  {
  case U_ARROW:
    if(curpos > 0) --curpos; break;
  case D_ARROW:
    if(curpos < NUM-1) ++curpos; break;
  case RETURN:
    printf(CLEAR);
    action(curpos); break;
  }
}}

/* display box around menu */
display_box()
{
```

```
  int i;
  printf(NORMAL);
  printf(ROWCOL,8,26);
  printf("<=======| RSM MENU |=======>");
  printf(ROWCOL,10,19);
  printf(ULCORNER); for (i=1; i<=41; i++) printf(HDOUBLE);
printf(URCORNER);
  printf(ROWCOL,14,19);
  printf(LLCORNER); for (i=1; i<=41; i++) printf(HDOUBLE);
printf(LRCORNER);
  for (i=1; i<=3; i++) {
    printf(ROWCOL,10+i,19);
    printf(VDOUBLE);
  }
  for (i=1; i<=3; i++) {
    printf(ROWCOL,10+i,61);
    printf(VDOUBLE);
  }
}

/* displays menu */
display(arr,msg,size,pos)
char *arr[];
char *msg[];
int size,pos;
{
  int j,result,center_pos;
  int r,c;
  for (j=0; j<size; j++) {
    r = 11+j; c = 20;
    printf(ROWCOL,r,c);   /* position cursor on screen */
    if (j == pos)
      printf(REVERSE);   /* print item in reverse */
    printf("%s\n",*(arr+j));  /* print the item */
    printf(NORMAL);       /* print in normal mode */
  }
  printf(BOTTOM);         /* cursor to lower left */
  printf(ERASE);
  result = strlen(*(msg+pos));
  center_pos = (80 - result)/2 ;
  printf("\x1B[24;%df",center_pos);  /* center message */
  printf("%s",*(msg+pos));  /* print message associated with menu
item */
}

/* gets the keyboard input */
getcode()
{
  int key,exit_while = 0;
  while (exit_while != 1) {
```

```
    key = getch();
    if (key == 0) {
      key = getch() ;     /* get the extend code */
      exit_while = 1;
    }
    else if (key == 13) exit_while = 1 ;  /* allow the carriage
return */
   }
  return( key ) ;    /* return key code value */
}

/* performs action based on cursor position */
action(pos)
int pos;
{
  switch(pos) {
    case 0:
      return_val =
_dos_findfirst(mas_filename,_A_NORMAL,&exe_file);
      if ( return_val == 0)
   return_val = system("master");
      else {
   printf(BOTTOM);
   printf(ERASE);
   printf("ERROR - THE FILE => %s <= DOES NOT EXIST IN DIRECTORY
<RETURN>"
    ,mas_filename);
   getch();
      }
      break;
    case 1:
      return_val =
_dos_findfirst(reg_filename,_A_NORMAL,&exe_file);
      if ( return_val == 0) {
   /* DELETE THE REGRESS.OUT FILE SO THAT IF AN ERROR OCCURS */
   /* IN THE REGRESSION -- THE REGOUT PROGRAM WILL NOT RUN */
   return_val = remove("regress.out");
   return_val = system("regress");
      }
      else {
   printf(BOTTOM);
   printf(ERASE);
   printf("ERROR - THE FILE => %s <= DOES NOT EXIST IN DIRECTORY
<RETURN>"
    ,reg_filename);
   getch();
      }
      return_val =
_dos_findfirst(out_filename,_A_NORMAL,&exe_file);
      if ( return_val == 0) {
```

```
   if ( _dos_findfirst("regress.out",_A_NORMAL,&exe_file) == 0)
     return_val = system("regout");
   else {
     printf(BOTTOM);
     printf(ERASE);
     printf("ERROR - THE REGRESSION DID NOT PRODUCE AN OUTPUT
FILE <RETURN>");
     getch();
   }
       }
       else {
   printf(BOTTOM);
   printf(ERASE);
   printf("ERROR - THE FILE => %s <= DOES NOT EXIST IN DIRECTORY
<RETURN>"
   ,out_filename);
   getch();
       }
       break;
      case 2:
       printf(CLEAR);
       exit(0);
   }
}
fscreen()
{
int i;
printf(CLEAR);
printf(BOLD);

/* print first row */
printf(ROWCOL,1,22);
printf(BLOCK UHALF UHALF UHALF BLOCK "  ");
printf(BLOCK UHALF UHALF UHALF BLOCK "  " BLOCK);
for (i=1; i<=3; i++) printf(UHALF);
printf(LHALF "  " BLOCK);
for (i=1; i<=4; i++) printf(UHALF);
printf("  "BLOCK BLOCK LHALF " " LHALF BLOCK BLOCK);

/* print the second row */
printf(ROWCOL,2,22);
printf(BLOCK "   " BLOCK "   " BLOCK);
printf(LEFTCOL,6);
printf(BLOCK "   " BLOCK "   ");
printf(BLOCK "      " BLOCK " " BLOCK LHALF BLOCK " " BLOCK);

/* print the third row */
printf(ROWCOL,3,22);
printf(BLOCK UHALF UHALF UHALF UHALF "  " BLOCK "      ");
printf(BLOCK UHALF BLOCK UHALF "    ");
```

```
printf(UHALF BLOCK LHALF "    " BLOCK " " UHALF);
printf(BLINK); printf("\x02"); printf(NORMAL); printf(BOLD);
printf(UHALF " " BLOCK);

/* print the fourth row */
printf(ROWCOL,4,22);
printf(BLOCK "       " BLOCK "       " BLOCK);
printf(" " UHALF BLOCK LHALF);
printf(LEFTCOL,5);
printf(UHALF BLOCK "  " BLOCK "     " BLOCK);

/* print the fifth row */
printf(ROWCOL,5,22);
printf(BLOCK "       " BLOCK LHALF LHALF LHALF BLOCK "   " BLOCK);
printf("   " BLOCK "  ");
for (i=1; i<=4; i++) printf(LHALF);
printf(BLOCK "  " BLOCK "     " BLOCK);
printf(NORMAL);

/* print the system name and authors */
printf(ROWCOL,7,25);
printf(ULCORNER);
for (i=1; i<=28; i++) printf(HDOUBLE);
printf(URCORNER);
printf(ROWCOL,8,25);
printf(VDOUBLE "  DECISION SUPPORT SYSTEM    " VDOUBLE);
printf(ROWCOL,9,25);
printf(VDOUBLE "            FOR              " VDOUBLE);
printf(ROWCOL,10,25);
printf(VDOUBLE "RESPONSE SURFACE METHODOLOGY" VDOUBLE);
printf(ROWCOL,11,25); printf(VDOUBLE); printf(ROWCOL,11,54);
printf(VDOUBLE);
printf(ROWCOL,12,25);
printf(VDOUBLE "            BY               " VDOUBLE);
printf(ROWCOL,13,25);
printf(VDOUBLE "    CAPT DAVID M. LEEPER     " VDOUBLE);
printf(ROWCOL,14,25);
printf(VDOUBLE "    CAPT GREGORY J. MEIDT    " VDOUBLE);
printf(ROWCOL,15,25);
printf(LLCORNER);
for (i=1; i<=28; i++) printf(HDOUBLE);
printf(LRCORNER);

/* print the information line at bottom */
printf(ROWCOL,18,8);
printf("IF YOU HAVE ANY QUESTIONS OR COMMENTS ABOUT THIS PROGRAM
CONTACT:\n\n");
printf("                CAPT DAVID M. LEEPER   PENTAGON/XOX1  AV
255-1535\n");
```

```
printf("                    CAPT GREGORY J. MEIDT  USAFA/DFM       AV
259-4310\n");
printf("                    MAJ KEN BAUER          AFIT/ENS        AV
785-3362\n");
printf("                    LT COL SKIP VALUSEK    AFIT/ENS        AV
785-2549\n\n");
printf("       ------------------- PRESS ANY KEY TO CONTINUE ");
printf("-------------------");
getch();
}
```

MASTER.EXE

```
*
* MASTER.PRG
*
CLEAR
init_consts()   &&  initialize all of the program constants
SET KEY F10 TO menu_line
level = 2
num_rows = 0
num_groups = 0
des_filename = SPACE(12)
res_filename = SPACE(12)
exit_now = .F.
SET MESSAGE TO 24 CENTER
DO WHILE !exit_now
  BEGIN SEQUENCE
  * DRAW THE OUTSIDE OUTBOX AND FILL MIDDLE OF BOX
  @ 1,1,23,79 BOX sl_box + chr(177)
  PUBLIC loc_array[5]
  PRIVATE s_locwin[5]
  num_levels = 1
  * UPDATE THE LOCATION BOX
  loc_array[num_levels] = menu_pad("MASTER MENU",22)
  s_locwin[num_levels] = current_loc(num_levels)
  prompt_size = 20

  DO box_top_msg WITH 9,29,14,50,box1,"RSM",.F.,norm,bright
  @ 24,0 SAY SPACE(80)
  @ 10,30 PROMPT menu_pad("Experimental Design",prompt_size);
          MESSAGE "SELECT TO EXECUTE EXPERIMENTAL DESIGN OPTION"
  @ 11,30 PROMPT menu_pad("Regression Inputs",prompt_size);
          MESSAGE "SELECT TO CREATE OR ADJUST REGRESSION INPUTS"
  @ 12,30 PROMPT menu_pad("Optimization",prompt_size);
          MESSAGE "OPTION CURRENTLY UNAVAILABLE"
  @ 13,30 PROMPT menu_pad("Quit",prompt_size);
          MESSAGE "SELECT TO QUIT TO THE DOS SYSTEM"
  choice = 1
  MENU TO choice
  DO CASE
    CASE choice = 1
      * UPDATE LOCATION BOX
      num_levels = num_levels + 1
      loc_array[num_levels] = menu_pad("EXPERIMENTAL DESIGN",22)
      s_locwin[num_levels] = current_loc(num_levels)

      exp_prompt_size = 26
      save_expwin = savescreen(14,26,19,53)
      exit_expdesign = .F.
      DO WHILE !exit_expdesign
```

```
      DO box_top_msg WITH 14,26,19,53,box1,"EXPERIMENTAL DESIGN",
;
         .F.,norm,bright
      SET MESSAGE TO 24 CENTER
      @ 24,0 SAY SPACE(80)
      @ 15,27 PROMPT menu_pad("Design Choices",exp_prompt_size);
              MESSAGE "SELECT TO CREATE THE LEVEL 2 OR 3 DESIGN
MATRIX"
      @ 16,27 PROMPT menu_pad("Factor Settings",exp_prompt_size);
              MESSAGE "SELECT TO INPUT THE LOW AND HIGH FACTOR
VALUES"
      @ 17,27 PROMPT menu_pad("Raw Data Matrix",exp_prompt_size);
              MESSAGE "SELECT TO CREATE THE RAW DATA MATRIX"
      @ 18,27 PROMPT menu_pad("Exit",exp_prompt_size);
              MESSAGE "SELECT TO EXIT"
      choice = 1
     MENU TO choice
     DO CASE    && CASE STRUCTURE FOR EXPERIMENTAL DESIGN
       CASE choice = 1      && DESIGN CHOICES

         * UPDATE LOCATION BOX
         num_levels = num_levels + 1
         loc_array[num_levels] = menu_pad("DESIGN CHOICES",22)
         s_locwin[num_levels] = current_loc(num_levels)

         DO p_design WITH level,num_rows,num_groups

         * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
         restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
         num_levels = num_levels - 1

       CASE choice = 2      && FACTOR SETTINGS

         * UPDATE LOCATION BOX
         num_levels = num_levels + 1
         loc_array[num_levels] = menu_pad("FACTOR SETTINGS",22)
         s_locwin[num_levels] = current_loc(num_levels)

         DO p_facset WITH level,num_rows

         * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
         restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
         num_levels = num_levels - 1

       CASE choice = 3      && RAW DATA MATRIX

         * UPDATE LOCATION BOX
         num_levels = num_levels + 1
         loc_array[num_levels] = menu_pad("RAW DATA MATRIX",22)
         s_locwin[num_levels] = current_loc(num_levels)
```

```
        DO p_rawdat WITH num_groups

         * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
        restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
        num_levels = num_levels - 1

      CASE choice = 4      &&  END EXPERIMENTAL DESIGN MENU
        exit_expdesign = .T.

      ENDCASE     && FOR EXPERIMENTAL DESIGN MENU OPTIONS
      ENDDO
      restscreen(14,26,19,53,save_expwin)

      * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
      restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
      num_levels = num_levels - 1

    CASE choice = 2
      * UPDATE LOCATION BOX
      num_levels = num_levels + 1
      loc_array[num_levels] = menu_pad("REGRESSION",22)
      s_locwin[num_levels] = current_loc(num_levels)

      reg_prompt_size = 26
      design_input   = .F.
      response_input = .F.
      exit_regdesign = .F.
      save_regwin = savescreen(14,26,20,53)
      DO WHILE !exit_regdesign
      DO box_top_msg WITH 14,26,20,53,box1,"REGRESSION", ;
        .F.,norm,bright
      @ 24,0 SAY SPACE(80)
      @ 15,27 PROMPT menu_pad("Design Matrix
Input",reg_prompt_size);
              MESSAGE "SELECT TO RETRIEVE A DESIGN MATRIX"
      @ 16,27 PROMPT menu_pad("Variable
selection",reg_prompt_size);
              MESSAGE "SELECT TO REDUCE THE NUMBER OF VARIABLES"
      @ 17,27 PROMPT menu_pad("Input Responses",reg_prompt_size);
              MESSAGE "SELECT TO INPUT RESPONSE DATA"
      @ 18,27 PROMPT menu_pad("Transformations",reg_prompt_size);
              MESSAGE "SELECT TO TRANSFORM THE RESPONSE DATA"
      @ 19,27 PROMPT menu_pad("Exit",reg_prompt_size);
              MESSAGE "SELECT TO EXIT"
      choice = 1
     MENU TO choice
     DO CASE    && CASE STRUCTURE FOR REGRESSION
       CASE choice = 1      && DESIGN MATRIX INPUT
```

```
          * UPDATE LOCATION BOX
          num_levels = num_levels + 1
          loc_array[num_levels] = menu_pad("DESIGN MATRIX
INPUT",22)
          s_locwin[num_levels] = current_loc(num_levels)

          @ 0,1 SAY "SELECT THE DESIGN FILE NAME <EXP.DES> IS
CURRENT"
          des_filename = get_filename("*.DES")
          @ 0,0 SAY SPACE(80)

     IF !EMPTY(des_filename)
       * make the input filename the current design file
       COPY FILE &des_filename TO exp.des
       design_input = .T.
     ENDIF

          * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
          restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
          num_levels = num_levels - 1

       CASE choice = 2      && VARIABLE SELECTION

          * UPDATE LOCATION BOX
          num_levels = num_levels + 1
          loc_array[num_levels] = menu_pad("VARIABLE
SELECTION",22)
          s_locwin[num_levels] = current_loc(num_levels)

          IF design_input
            DO p_varsel WITH des_filename
          ELSE
            @ 0,1 SAY "ERROR -> NO DESIGN INPUT <RETURN>"
            wait_key = inkey(0)
            @ 0,0 SAY SPACE(80)
          ENDIF

          * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
          restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
          num_levels = num_levels - 1

       CASE choice = 3      && RESPONSE INPUT

          * UPDATE LOCATION BOX
          num_levels = num_levels + 1
          loc_array[num_levels] = menu_pad("RESPONSE INPUT",22)
          s_locwin[num_levels] = current_loc(num_levels)

          DO p_response
     IF !EMPTY(res_filename)
```

```
      response_input = .T.
    ENDIF

         * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
        restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
        num_levels = num_levels - 1

      CASE choice = 4      && RESPONSE TRANSFORMATIONS

        * UPDATE LOCATION BOX
        num_levels = num_levels + 1
        loc_array[num_levels] = menu_pad("TRANSFORMATIONS",22)
        s_locwin[num_levels] = current_loc(num_levels)

        IF response_input
          DO p_transform
        ELSE
          @ 0,1 SAY "ERROR -> NO RESPONSE FILE INPUT <RETURN>"
          wait_key = inkey(0)
          @ 0,0 SAY SPACE(80)
        ENDIF

         * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
        restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
        num_levels = num_levels - 1

  CASE choice = 5      &&  END REGRESSION MENU
        exit_regdesign = .T.

     ENDCASE     && FOR REGRESSION MENU OPTIONS
     ENDDO
     restscreen(14,26,20,53,save_regwin)

     * RESTORE SCREEN FOR LOCATION BOX NEXT LEVEL UP
     restscreen(2,55,3+num_levels,78,s_locwin[num_levels])
     num_levels = num_levels - 1

  CASE choice = 3
  CASE choice = 4
    exit_now = .T.
  ENDCASE  && FOR MASTER MENU OPTIONS
  END  && END FOR SEQUENCE STRUCTURE
ENDDO
SET CURSOR ON
CLEAR
RETURN

PROCEDURE help
PARAMETERS proc,line,var
  SET KEY F1 TO
```

```
  SAVE SCREEN
  @ 0,0  SAY SPACE(80)
  @ 24,0 SAY SPACE(80)
  @ 0,1 SAY "USE THE ARROWS AND PAGE UP/DOWN TO MOVE INSIDE THE
HELP WINDOW"
  IF FILE("pcrsm.hlp")
    @ 1,1,23,79 BOX sl_box + chr(177)
    @ 8,10 SAY "HELP INFORMATION FOR SYSTEM -- PRESS ESC TO
LEAVE"
    @ 9,1,19,79 BOX SL_BOX
    MEMOEDIT(MEMOREAD("pcrsm.hlp"),10,2,18,78,.F.)
  ELSE
    @ 0,0 SAY SPACE(80)
    @ 0,1 SAY "ERROR -- HELP FILE -> PCRSM.HLP <- NOT AVAILABLE
<RETURN>"
    wait_key = inkey(0)
  ENDIF
  RESTORE SCREEN
SET KEY F1 TO help
RETURN


PROCEDURE menu_line
PARAMETERS proc,line,var
SET KEY F10 TO
SAVE SCREEN
@ 0,0  SAY SPACE(80)
@ 24,0 SAY SPACE(80)
@ 0,01 PROMPT "Master" ;
       MESSAGE "SELECT TO GET THE MASTER MENU"
@ 0,09 PROMPT "Hook book" ;
       MESSAGE "SELECT TO WRITE DOWN SYSTEM ENHANCEMENTS"
@ 0,30 PROMPT "Notepad" ;
       MESSAGE "SELECT TO WRITE DOWN ANY NOTES"
choice = 1
MENU TO choice
DO CASE
  CASE choice = 1
    IF var = "CHOICE"
      @ 0,0  SAY SPACE(80)
      SET KEY F10 TO menu_line
      CLEAR
      KEYBOARD CHR(ENTER)
      BREAK
    ELSE
      @ 0,1 SAY "ERROR -- ONLY FROM A MENU => PRESS ESC TO GET TO
MENU <RETURN>"
      wait_key = inkey(0)
      RESTORE SCREEN
    ENDIF
```

```
  CASE choice = 2
    @ 1,1,23,79 BOX sl_box + chr(177)
    DO p_hook WITH .T.
    RESTORE SCREEN
  CASE choice = 3
    @ 1,1,23,79 BOX sl_box + chr(177)
    DO p_hook WITH .F.
    RESTORE SCREEN
ENDCASE
@ 0,0  SAY SPACE(80)
SET KEY F10 TO menu_line
RETURN

PROCEDURE box_top_msg
PARAM t,l,b,r,box_str,str,intense,norm,bright
PRIVATE l_spaces,r_spaces,msg_len

@ t,l,b,r BOX box_str
msg_len= len(str)
l_spaces = int(((r-l-1)-msg_len)/2)
r_spaces = l_spaces
* adjust left side if not equal
IF r_spaces + l_spaces + msg_len != (r-l-1)
  l_spaces = l_spaces + 1
ENDIF
@ t,l SAY substr(box_str,1,1)
@ t,l+1 SAY replicate(substr(box_str,2,1),l_spaces)
IF intense
  SET COLOR TO &bright
  @t,l+l_spaces+1 SAY str
  SET COLOR TO &norm
ELSE
  @ t,l+l_spaces+1 SAY str
ENDIF
@ t,l+l_spaces + 1 + len(str) SAY ;
    replicate(substr(box_str,2,1),r_spaces)
@ t,r SAY substr(box_str,3,1)
RETURN

FUNCTION current_loc
PARAMETERS num_levels
PRIVATE save_locwin
save_locwin = savescreen(2,55,3+num_levels,78)
DO box_top_msg WITH 2,55,3+num_levels,78,box2,"CURRENT LOCATION",
;
                    .F.,norm,bright
FOR i = 1 to num_levels
  IF i = num_levels
    SET COLOR TO I
    @ 2+i,56 SAY loc_array[i]
```

```
    SET COLOR TO
  ELSE
    @ 2+i,56 SAY loc_array[i]
  ENDIF
NEXT
RETURN (save_locwin)
```

DESIGN.PRG

```
*
*  DESIGN.PRG
*
PROCEDURE p_design
PARAMETERS level,num_rows,num_groups
BEGIN SEQUENCE
          PRIVATE ar[20],desarr[8]
          IF FILE("exp.des")
             ERASE exp.des
          ENDIF
          num_center_pts = 0
          num_reps = 0
          num_rows = 0
          save_reswin = savescreen(14,10,20,42)
          SET KEY ESC TO proc_esc
          info_msg = "THE NUMBER OF FACTORS HELP DETERMINE THE
MODEL"
          @ 0,1 SAY info_msg
          @ 24,1 SAY "Input the number of FACTORS" + ;
                   " [ENTER 0 FOR THREE-LEVEL MODELS] =>" ;
               GET num_rows VALID validate(1,info_msg)
        READ
        @ 0,0 SAY SPACE(80)
        @ 24,0 SAY SPACE(80)
          IF num_rows = 0    && THREE-LEVEL MODELS APPLY
            size = 3         && NUMBER OF ITEMS TO DISPLAY
            ar[1] = "FULL FACTORIAL"
            ar[2] = "CCD"
            ar[3] = "BOX-BEHNKEN"
            level = 3
            orthogonal = 0
            alpha = 0.00000
            @ 14,10 SAY "THREE - LEVEL MODELS" + SPACE(14)
            SET KEY ESC TO ESC
            sel = abrowse(ar,size,ENTER,15,10,20,42)
           IF sel = 0    && IF ESCAPE KEY PRESSED THEN EXIT
             restscreen(14,10,20,42,save_reswin)
             RETURN
           ENDIF
            SET KEY ESC TO proc_esc
           num_rows = 2
```

```
            info_msg = "THIS IS THE ACTUAL NUMBER OF FACTORS " +
;
                       "FOR THE 3-LEVEL DESIGNS"
            @ 0,1 SAY info_msg
           @ 24,1 SAY "Input the number of FACTORS =>" ;
                GET num_rows VALID validate(2,info_msg)
           READ
           @ 0,0 SAY SPACE(80)
           @ 24,0 SAY SPACE(80)

            IF sel < 3   && OTHER THAN BOX-BEHNKEN DESIGN
              * MUST HAVE AT LEAST ONE CENTER POINT FOR CCD
OPTION
              IF sel = 2
                num_center_pts = 1
              ENDIF
              info_msg = "PROVIDES INFORMATION ABOUT " + ;
                         "THE MIDDLE OF THE DESIGN"
              @ 0,1 SAY info_msg
              @ 24,1 SAY "Input the number of CENTER POINTS " + ;

                       "<RETURN FOR NONE> =>" GET num_center_pts
;
                   VALID validate(3,info_msg)
              READ
              @ 0,0 SAY SPACE(80)
             @ 24,0 SAY SPACE(80)

              info_msg = "ONE REPLICATION --> REPLICATES THE
ENTIRE DESIGN"
              @ 0,1 SAY info_msg
             @ 24,1 SAY "Input the number of REPLICATIONS " + ;
                       "<RETURN FOR NONE> =>" GET num_reps ;
                   VALID validate(4,info_msg)
             READ
             @ 0,0 SAY SPACE(80)
             @ 24,0 SAY SPACE(80)

              IF sel = 2    && CCD DESIGN
                SET FIXED ON
                alpha = calc_alpha(num_rows,num_center_pts)
                SET FIXED OFF
                compare_alpha = alpha
                @ 0,1 SAY "ALPHA VALUE CONTROLS WHETHER OR NOT "
+;
                         "DESIGN IS ORTHOGONAL"
               @ 24,1 SAY "If factors can be set to " + ;
                         "orthogonal ALPHA - <RETURN> " + ;
                         "else <VALUE> " GET alpha  PICT
"99.99999"
```

```
                    READ
                    @ 0,0 SAY SPACE(80)
                    @ 24,0 SAY SPACE(80)
                    IF alpha = compare_alpha
                      orthogonal = 1
                    ENDIF
                  ENDIF
                ENDIF
*  BUILD THE 3-LEVEL DESIGN CREATION PARAMETER ARRAY
                desarr[1] = num_groups
                desarr[2] = INT(num_rows)
                desarr[3] = level
                desarr[4] = sel
                desarr[5] = orthogonal
                desarr[6] = alpha
                desarr[7] = INT(num_center_pts)
                desarr[8] = num_reps
 @ 0,0 SAY SPACE(80)
 @ 0,0 SAY ltrim(str(desarr[1])) + " ¦ " + ltrim(str(desarr[2]))
+ ;
 " ¦ " + ltrim(str(desarr[3])) + " ¦ " + ltrim(str(desarr[4])) +
;
 " ¦ " + ltrim(str(desarr[5])) + " ¦ " + ltrim(str(desarr[6])) +
;
 " ¦ " + ltrim(str(desarr[7])) + " ¦ " + ltrim(str(desarr[8]))
 WAIT_K = INKEY(0)
 @ 0,0 SAY SPACE(80)

              ELSE      &&  TWO - LEVEL DESIGNS
               DECLARE res[4],runs[4]
               afill(res,0)
               afill(runs,0)
                 num_elements = 1

* GET THE DESIGN'S RESOLUTION AND NUMBER OF RUNS BASED ON LEVEL &
# FACTORS
                DO p_getres WITH level,num_rows,res,runs,num_elements


                ar[1] = "   GROUP SCREENING"  && ADD OPTION TO LIST

                FOR i = 1 TO num_elements
                  IF res[i] = 0
                    ar[i+1] = "   FULL FACTORIAL   " + STR(runs[i],3)
                  ELSE
                    ar[i+1] = SPACE(3) + STR(res[i],1) + SPACE(16) ;
                              + STR(runs[i],3)
                  ENDIF
                NEXT
                @ 14,10 SAY "   RES              # RUNS" + SPACE(18)
```

```
IF num_rows >= 12
  @ 0,1 SAY "BECAUSE THE NUMBER OF FACTORS IS GREATER
THAN" ;
           + " 11 - SUGGEST GROUP SCREENING"
ENDIF

size = num_elements + 1
SET KEY ESC TO ESC
sel = abrowse(ar,size,ENTER,15,10,20,42)
IF sel = 0   && IF ESCAPE KEY PRESSED THEN EXIT
 restscreen(14,10,20,42,save_reswin)
 RETURN
ENDIF
SET KEY ESC TO proc_esc

@ 0,0 SAY SPACE(80)

*  based on the "sel" value get the proper resolution
* to pass to the routine to get the value
* check to see if group screening was selected sel =
1
level = 2
num_groups = 0
IF sel = 1    && GROUP SCREENING
  num_groups = 1
  info_msg = "ASSISTS IN REDUCING THE FACTOR SPACE"
  @ 0,1 SAY info_msg
 @ 24,1 SAY "Input the number of GROUPS " + ;
           "<RETURN FOR NONE> =>" GET num_groups ;
        VALID validate(5,info_msg)
  READ
  @ 0,0 SAY SPACE(80)
 @ 24,0 SAY SPACE(80)
ENDIF
info_msg = "PROVIDES INFORMATION ABOUT " + ;
           "THE MIDDLE OF THE DESIGN"
@ 0,1 SAY info_msg
@ 24,1 SAY "Input the number of CENTER POINTS " + ;
         "<RETURN FOR NONE> =>" GET num_center_pts ;
      VALID validate(3,info_msg)
READ
@ 0,0 SAY SPACE(80)
@ 24,0 SAY SPACE(80)

info_msg = "ONE REPLICATION --> REPLICATES THE ENTIRE
DESIGN"
@ 0,1 SAY info_msg
@ 24,1 SAY "Input the number of REPLICATIONS " + ;
         "<RETURN FOR NONE> =>" GET num_reps ;
      VALID validate(4,info_msg)
```

```
         READ
         @ 0,0 SAY SPACE(80)
         @ 24,0 SAY SPACE(80)

           *  BUILD THE DESIGN CREATION PARAMETER ARRAY
           desarr[1] = num_groups
           desarr[2] = num_rows
           desarr[3] = level
           IF num_groups > 0
             * IF GROUP SCREENING THEN USE LOWEST RESOLUTION
DESIGN <3>
             DO CASE
               CASE num_groups = 2
                  resolution = 0
                CASE num_groups = 4
                  resolution = 4
                OTHERWISE
                  resolution = 3
             ENDCASE
             desarr[4] = resolution
           ELSE
             * sel one less because group screening is first
option
             IF sel > 1
               desarr[4] = res[sel-1]
             ENDIF
           ENDIF
           desarr[5] = num_center_pts
           desarr[6] = num_reps
 @ 0,0 SAY SPACE(80)
 @ 0,0 SAY ltrim(str(desarr[1])) + " ¦ " + ltrim(str(desarr[2]))
+ ;
 " ¦ " + ltrim(str(desarr[3])) + " ¦ " + ltrim(str(desarr[4])) +
;
 " ¦ " + ltrim(str(desarr[5])) + " ¦ " + ltrim(str(desarr[6]))
 WAIT_K = INKEY(0)
 @ 0,0 SAY SPACE(80)
         ENDIF
         SET CURSOR ON
*  CALL THE C CODE FUNCTION TO BUILD THE DESIGN
         SAVE SCREEN
         CLEAR
*? "THIS IS WHERE THE C CODE TAKES OVER  -- PRESS RETURN"
*wait_key = inkey(0)
         cdesign(desarr)
         RESTORE SCREEN
         restscreen(14,10,20,42,save_reswin)
         IF FILE("exp.des")
           overwrite = .F.
```

```
            prompt_msg = "PLEASE ENTER THE DESIGN MATRIX
FILENAME"
            des_file_name =
input_filename(".DES",@overwrite,prompt_msg)
            IF !EMPTY(des_file_name) .AND. overwrite
              COPY FILE exp.des TO &des_file_name
            ENDIF
          ENDIF
END   && FOR END SEQUENCE
SET KEY ESC TO ESC
RETURN

PROCEDURE proc_esc
  @ 0,0 SAY SPACE(80)
  restscreen(14,10,20,42,save_reswin)
  BREAK
RETURN

FUNCTION calc_alpha
PARAMETERS num_factors,center_pts
PRIVATE num_factors,center_pts,F,T,Q
  SET DECIMALS TO 5
  F = 2**num_factors  && NUM OF ROWS IN THE CORE CCD DESIGN
MATRIX
  T = (2 * num_factors) + center_pts
  Q = (SQRT(F + T) - SQRT(F))**2
RETURN(((Q * F)/4)**(1/4))

FUNCTION validate
PARAMETERS valid_num,info_msg
  DO CASE
    CASE valid_num = 1
      IF ((num_rows = 0).OR.(num_rows > 1))
        RETURN(.T.)
      ENDIF
      valid_msg = "FACTORS MUST BE 0, 2 OR MORE"
    CASE valid_num = 2
      IF ((num_rows >= 2).AND.(num_rows <= 6))
        RETURN(.T.)
      ENDIF
      valid_msg = "FACTORS MUST BE BETWEEN 2 AND 6"
    CASE valid_num = 3
      IF ((sel = 2).AND.(level = 3))
        compare_val = 1
        valid_msg = "CENTER POINTS MUST BE 1 OR GREATER FOR CCD"
      ELSE
        compare_val = 0
        valid_msg = "CENTER POINTS CANNOT BE NEGATIVE"
      ENDIF
      IF num_center_pts >= compare_val
```

```
        RETURN(.T.)
      ENDIF
    CASE valid_num = 4
      IF num_reps >= 0
        RETURN(.T.)
      ENDIF
      valid_msg = "REPLICATIONS CANNOT BE NEGATIVE"
    CASE valid_num = 5
      IF ((num_groups >= 1).AND.(num_groups <= num_rows))
        RETURN(.T.)
      ENDIF
      valid_msg = "# OF GROUPS MUST BE BETWEEN ONE AND # OF
FACTORS"
  ENDCASE
  valid_msg = "ERROR -> " + valid_msg + " <PRESS RETURN>"
  @ 0,1 SAY valid_msg + SPACE(80 - len(valid_msg))
  wait_key = inkey(0)
  info_msg = info_msg + " <ENTER NEW VALUE>"
  @ 0,1 SAY info_msg + SPACE(80 - len(info_msg))
RETURN(.F.)
```

<u>GETRES.PRG</u>

```
PROCEDURE p_getres
PARAMETERS level,num_rows,res,runs,num_elements
  DO CASE
    CASE num_rows = 2
      res[1]  = 0
      runs[1] = 4
    CASE num_rows = 3
      num_elements = 2
      res[1]  = 3
      res[2]  = 0
      runs[1] = 4
      runs[2] = 8
    CASE num_rows = 4
      num_elements = 2
      res[1]  = 4
      res[2]  = 0
      runs[1] = 8
      runs[2] = 16
    CASE num_rows = 5
      num_elements = 3
      res[1]  = 3
      res[2]  = 5
      res[3]  = 0
      runs[1] = 8
      runs[2] = 16
      runs[3] = 32
```

```
CASE num_rows = 6
  num_elements = 4
  res[1]  = 3
  res[2]  = 4
  res[3]  = 6
  res[4]  = 0
  runs[1] = 8
  runs[2] = 16
  runs[3] = 32
  runs[4] = 64
CASE num_rows = 7
  num_elements = 3
  res[1]  = 3
  res[2]  = 4
  res[3]  = 7
  runs[1] = 8
  runs[2] = 16
  runs[3] = 64
CASE num_rows = 8
  num_elements = 3
  res[1]  = 3
  res[2]  = 4
  res[3]  = 5
  runs[1] = 8
  runs[2] = 16
  runs[3] = 64
CASE (num_rows >= 9).AND.(num_rows <= 11)
  num_elements = 2
  res[1]  = 3
  res[2]  = 4
  runs[1] = 16
  runs[2] = 32
CASE num_rows = 12
  res[1]  = 3
  runs[1] = 12
CASE (num_rows >= 13).AND.(num_rows <= 16)
  res[1]  = 3
  runs[1] = 16
CASE (num_rows >= 17).AND.(num_rows <= 20)
  res[1]  = 3
  runs[1] = 20
CASE (num_rows >= 21).AND.(num_rows <= 24)
  res[1]  = 3
  runs[1] = 24
CASE (num_rows >= 25).AND.(num_rows <= 28)
  res[1]  = 3
  runs[1] = 28
CASE (num_rows >= 29).AND.(num_rows <= 32)
  res[1]  = 3
  runs[1] = 32
```

```
    CASE (num_rows >= 33).AND.(num_rows <= 36)
      res[1]  = 3
      runs[1] = 36
    CASE (num_rows >= 37).AND.(num_rows <= 40)
      res[1]  = 3
      runs[1] = 40
  ENDCASE
RETURN
```

CREATDES.C

```
/*  PROGRAM CREATDES.C  --  A FUNCTION FOR CLIPPER TO DETERMINE
    WHICH DESIGN PROGRAM TO EXECUTE BASED ON LEVEL AND PASS DATA
*/

# include "nandef.h"
# include "extend.h"

float alpha,desmat[65][130];
int gpscreen,factor,level;
int res,choice,orthog;
int ceps,reps;

CLIPPER cdesign()
{
  gpscreen = _parni(1,1);
  factor   = _parni(1,2);
  level    = _parni(1,3);
  if (level == 2) {
    res  = _parni(1,4);
    ceps = _parni(1,5);
    reps = _parni(1,6);
    cdesign2(); }          /*  call the create level-2 design
function */
  else {
    choice = _parni(1,4);
    orthog = _parni(1,5);
    alpha  = _parnd(1,6);
    ceps   = _parni(1,7);
    reps   = _parni(1,8);
    cdesign3();            /*  call the create level-3 design
function */
   }
}  /* end of cdesign function */
```

LEVEL2.C

```
/* THIS PROGRAM GENERATES TWO LEVEL DESIGN MATRICES */

# include <stdio.h>
# include <math.h>

extern float alpha,desmat[130][65];
extern int gpscreen,factor,level;
extern int res,choice,orthog;
extern int ceps,reps;

/* MAIN DETERMINES # FACTORS, RESOLUTION, AND DESIGN */
cdesign2()
{

FILE *outfile;
int row,cols,totrow,desfac;
int i,j,k,l;
int c;     /* # DIFFERENT ROWS */
int gsfactor;  /* NUMBER OF TOTAL FACTORS FOR GROUP SCREENING */
int pb;
char ch = 's';
char wait_key;

orthog = 1;
choice = 0;
pb = 0;  /* TELLS WHETHER OR NOT PLACKETT-BURMAN DESIGN */

if (gpscreen != 0)
  {
  gsfactor = factor;  /* NUMBER OF TOTAL FACTORS FOR THE GROUPS
*/
  factor   = gpscreen;  /* ACTUAL NUMBER OF FACTORS FOR GROUP
SCREENING */
  }

/* GENERATE 2 LEVEL [1-,1] DESIGN MATRIX */

 if (level == 2)
  {

   if (factor == 2)

     /* 2**2 full factorial */
     { desfac = 2;
       row = pow(2,desfac)*(reps+1);
       cols = pow(2,desfac);
       totrow = row + ceps*(reps+1);
       design(desfac,row,ceps,totrow);
```

```
      lev2_label(desfac);
    }

  else if (factor == 3)

     { if (res == 3)

        /* 2**(3-1) res III design, 3=12 */
        { desfac = 2;
          row = pow(2,desfac)*(reps+1);
          cols = pow(2,desfac);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][4] = 3;
        }

      else

      /* 2**3 full factorial */
        { desfac = 3;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
        }
    }

  else if (factor == 4)

    { if (res == 4)

        /* 2**(4-1) res IV design, 4=123 */
        { desfac = 3;
          res = 4;
          cols = pow(2,desfac);
          row = pow(2,3)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][8] = 4;
        }
      else

      /* 2**4 full factorial */
        { desfac = 4;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
```

```
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
        }
    }

  else if (factor == 5)

    { if (res == 3)

        /* 2**(5-2) res III design, 4=12, 5=13 */
        { desfac = 3;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][5] = 4;
          desmat[0][6] = 5;
        }

      else if (res == 5)

        /* 2**(5-1) res V design, 5=1234 */
        { desfac = 4;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][16] = 5;
        }

      else

      /* 2**5 full factorial */
        { desfac = 5;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
        }
    }

  else if (factor == 6)

    { if (res == 3)

        /* 2**(6-3) res III design, 4=12, 5=13, 6=23 */
        { desfac = 3;
```

```
      cols = pow(2,desfac);
      row = pow(2,desfac)*(reps+1);
      totrow = row + ceps*(reps+1);
      design(desfac,row,ceps,totrow);
      lev2_label(desfac);
      desmat[0][5] = 4;
      desmat[0][6] = 5;
      desmat[0][7] = 6;
    }

  else if (res == 4)

    /* 2**(6-2) res IV design, 5=123, 6=234 */
    { desfac = 4;
      cols = pow(2,desfac);
      row = pow(2,desfac)*(reps+1);
      totrow = row + ceps*(reps+1);
      design(desfac,row,ceps,totrow);
      lev2_label(desfac);
      desmat[0][9] = 5;
      desmat[0][15] = 6;
    }

  else if (res == 5)

       printf("DESIGN NOT AVAILABLE\n");

  else if (res == 6)

    /* 2**(6-1) res VI design, 6=12345 */
    { desfac = 5;
      cols = pow(2,desfac);
      row = pow(2,desfac)*(reps+1);
      totrow = row + ceps*(reps+1);
      design(desfac,row,ceps,totrow);
      lev2_label(desfac);
      desmat[0][32] = 6;
}

  else

  /* 2**6 full factorial */
    { desfac = 6;
      cols = pow(2,desfac);
      row = pow(2,desfac)*(reps+1);
      totrow = row + ceps*(reps+1);
      design(desfac,row,ceps,totrow);
      lev2_label(desfac);
    }
 }
```

```
else if (factor == 7)

  { if (res == 3)

      /* 2**(7-4) res III design, 4=12, 5=13, 6=23, 7=123 */
      { desfac = 3;
        cols = pow(2,desfac);
        row = pow(2,desfac)*(reps+1);
        totrow = row + ceps*(reps+1);
        design(desfac,row,ceps,totrow);
        lev2_label(desfac);
        desmat[0][5] = 4;
        desmat[0][6] = 5;
        desmat[0][7] = 6;
        desmat[0][8] = 7;
      }

    else if (res == 4)

      /* 2**(7-3) res IV design, 5=123, 6=234, 7=134 */
      { desfac = 4;
        cols = pow(2,desfac);
        row = pow(2,desfac)*(reps+1);
        totrow = row + ceps*(reps+1);
        design(desfac,row,ceps,totrow);
        lev2_label(desfac);
        desmat[0][9] = 5;
        desmat[0][15] = 6;
        desmat[0][14] = 7;
      }

    else if (res == 5)

       printf("DESIGN NOT AVAILABLE\n");

    else if (res == 6)

       printf("DESIGN NOT AVAILABLE\n");

    else if (res == 7)

      /* 2**(7-1) res VII design, 7=123456 */
      { desfac = 6;
        cols = pow(2,desfac);
        row = pow(2,desfac)*(reps+1);
        totrow = row + ceps*(reps+1);
        design(desfac,row,ceps,totrow);
        lev2_label(desfac);
        desmat[0][64] = 6;
      }
```

```
    else  printf("2**7 DESIGN UNAVAILABLE\n");

    /* 2**7 full factorial - UNAVAILABLE
      { desfac = 7;
        cols = pow(2,6);
        row = pow(2,desfac)*(reps+1);
        totrow = row + ceps*(reps+1);
        design(desfac,row,ceps,totrow);
        lev2_label(desfac);
      }                                   */
  }

else if (factor == 8)

  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
   row = 12;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 9;
   pb = 1;
       }

     else if (res == 4)

       /* 2**(8-4) res IV design, 5=234, 6=134, 7=123, 8=124 */
       { desfac = 4;
         cols = pow(2,desfac);
         row = pow(2,desfac)*(reps+1);
         totrow = row + ceps*(reps+1);
         design(desfac,row,ceps,totrow);
         lev2_label(desfac);
         desmat[0][15] = 5;
         desmat[0][14] = 6;
         desmat[0][9] = 7;
         desmat[0][12] = 8;
       }

     else if (res == 5)

       /* 2**(8-2) res V design, 7=1234, 8=1256 */
       { desfac = 6;
         cols = pow(2,desfac);
         row = pow(2,desfac)*(reps+1);
         totrow = row + ceps*(reps+1);
```

```
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][18] = 7;
          desmat[0][52] = 8;
        }

      else if (res == 6 || res == 7)

         printf("DESIGN NOT AVAILABLE\n");

      else if (res == 8)  printf("2**(8-1) DESIGN
UNAVAILABLE\n");


        /* 2**(8-1) res VIII design, 8=1234567
        { desfac = 7;
          cols = pow(2,6);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
        }                                           */

      else

       printf("FULL FACTORIAL NOT AVAILABLE - 256 RUNS\n");

    }

  else if (factor == 9)

    { if (res == 3)

      /* 2**(9-5) res III design, 5=123, 6=234, 7=134, 8=124,
9=1234 */
        { desfac = 4;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][9] = 5;
          desmat[0][15] = 6;
          desmat[0][14] = 7;
          desmat[0][12] = 8;
          desmat[0][16] = 9;
        }

      else if (res == 4)
```

```
      /* 2**(9-4) res IV design, 6=2345, 7=1345, 8=1245, 9=1235
*/
        { desfac = 5;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][31] = 6;
          desmat[0][30] = 7;
          desmat[0][28] = 8;
          desmat[0][24] = 9;
        }

      else if (res == 5)

         printf("DESIGN NOT AVAILABLE\n");

      else if (res == 6)  printf("2**(9-2) DESIGN
UNAVAILABLE\n");

        /* 2**(9-2) res VI design, 8=13467 9=23567
        { desfac = 7;
          cols = pow(2,6);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
        }                                                    */

      else if (res == 7 || res == 8)

         printf("DESIGN NOT AVAILABLE\n");


      else

        printf("FULL FACTORIAL NOT AVAILABLE\n");

    }

  else if (factor == 10)

    { if (res == 3)

      /* 2**(10-6) res III design, 5=123, 6=234, 7=134,
                                   8=124, 9=1234, 10=12 */
        { desfac = 4;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
```

```
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][9] = 5;
          desmat[0][15] = 6;
          desmat[0][14] = 7;
          desmat[0][12] = 8;
          desmat[0][16] = 9;
          desmat[0][6] = 10;
        }

      else if (res == 4)

     /* 2**(10-5) res IV design, 6=1234, 7=1235, 8=1245, 9=1345
                                         10=2345 */
        { desfac = 5;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][17] = 6;
          desmat[0][24] = 7;
          desmat[0][28] = 8;
          desmat[0][30] = 9;
          desmat[0][31] = 10;
        }

      else if (res == 5)   printf("2**(10-3) DESIGN
UNAVAILABLE\n");

      /* 2**(10-3) res V design, 8=1237 9=2345 10=1346
        { desfac = 7;
          cols = pow(2,6);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
        }                                                      */

      else

       printf("DESIGN NOT AVAILABLE\n");

    }

  else if (factor == 11)

    { if (res == 3)
```

```
      /* 2**(11-7) res III design, 5=123, 6=234, 7=134, 8=124,
9=1234
                                                10=12  11=13 */
        { desfac = 4;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][9] = 5;
          desmat[0][15] = 6;
          desmat[0][14] = 7;
          desmat[0][12] = 8;
          desmat[0][16] = 9;
          desmat[0][6] = 10;
          desmat[0][7] = 11;
        }

      else if (res == 4)

        /* 2**(11-6) res IV design, 6=123, 7=234, 8=345, 9=134
                                                 10=145 11=245 */
        { desfac = 5;
          cols = pow(2,desfac);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
          desmat[0][10] = 6;
          desmat[0][16] = 7;
          desmat[0][29] = 8;
          desmat[0][15] = 9;
          desmat[0][26] = 10;
          desmat[0][27] = 11;
        }

      else if (res == 5)   printf("2**(11-4) DESIGN
UNAVAILABLE\n");

      /* 2**(11-4) res V design, 8=1237 9=2345 10=1346
11==1234567
        { desfac = 7;
          cols = pow(2,6);
          row = pow(2,desfac)*(reps+1);
          totrow = row + ceps*(reps+1);
          design(desfac,row,ceps,totrow);
          lev2_label(desfac);
        }
 */
```

```
    else  printf("DESIGN NOT AVAILABLE\n");

  }

else if (factor == 12)

  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {row = 16;
   cols = row;
   orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 13;
        }
  }
else if (factor == 13)

  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
         row = 16;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 14;}
    else  printf("DESIGN NOT AVAILABLE\n");
  }
else if (factor == 14)

  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
         row = 16;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 15;}
    else  printf("DESIGN NOT AVAILABLE\n");
  }
else if (factor == 15)

  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
```

```
  {
   row = 16;
   cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
   cols = 16;}
     else  printf("DESIGN NOT AVAILABLE\n");
   }
else if (factor == 16)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
  {
        row =  20;
   cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
   cols = 17;}
     else  printf("DESIGN NOT AVAILABLE\n");
   }
else if (factor == 17)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
  {
        row = 20;
   cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
   cols = 18;}
     else  printf("DESIGN NOT AVAILABLE\n");
   }
else if (factor == 18)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
  {
        row = 20;
   cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
   cols = 19;}
     else  printf("DESIGN NOT AVAILABLE\n");
```

```
  }
else if (factor == 19)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
  {
   row = 20;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 20;}
    else  printf("DESIGN NOT AVAILABLE\n");
  }
else if (factor == 20)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
  {
         row = 24;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 21;}
    else  printf("DESIGN NOT AVAILABLE\n");
  }
else if (factor == 21)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
  {
         row = 24;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 22;}
    else  printf("DESIGN NOT AVAILABLE\n");
  }
else if (factor == 22)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
  {
         row = 24;
```

```
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 23;}
     else  printf("DESIGN NOT AVAILABLE\n");
   }
else if (factor == 23)

   { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
   row = 24;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 24;}
     else  printf("DESIGN NOT AVAILABLE\n");
   }
else if (factor == 24)

   { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
         row = 32;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 25;}
     else  printf("DESIGN NOT AVAILABLE\n");
   }

else if (factor == 25)

   { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
         row = 32;
   cols = row;
         orthog = 0;
         totrow = row + ceps*(reps+1);
         plackett(row,cols,ceps,totrow);
   cols = 26;}
     else  printf("DESIGN NOT AVAILABLE\n");
   }
```

```
else if (factor == 26)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
 {
        row = 32;
  cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
  cols = 27;}
    else  printf("DESIGN NOT AVAILABLE\n");
  }
else if (factor == 27)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
 {
        row = 32;
  cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
  cols = 28;}
    else  printf("DESIGN NOT AVAILABLE\n");
  }
else if (factor == 28)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
 {
        row = 32;
  cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
  cols = 29;}
    else  printf("DESIGN NOT AVAILABLE\n");
  }

else if (factor == 29)

  { if (res == 3)

       /* PLACKETT-BURMAN DESIGN  */
 {
  row = 32;
```

```
      cols = row;
            orthog = 0;
            totrow = row + ceps*(reps+1);
            plackett(row,cols,ceps,totrow);
      cols = 30;}
        else  printf("DESIGN NOT AVAILABLE\n");
      }
   else if (factor == 30)

      { if (res == 3)

           /* PLACKETT-BURMAN DESIGN  */
     {
            row = 32;
      cols = row;
            orthog = 0;
            totrow = row + ceps*(reps+1);
            plackett(row,cols,ceps,totrow);
      cols = 31;}
        else  printf("DESIGN NOT AVAILABLE\n");
      }
   else if (factor == 31)

      { if (res == 3)

           /* PLACKETT-BURMAN DESIGN  */
     {
      row = 32;
      cols = row;
            orthog = 0;
            totrow = row + ceps*(reps+1);
            plackett(row,cols,ceps,totrow);
      cols = 32;}
        else  printf("DESIGN NOT AVAILABLE\n");
      }
   else if (factor == 32)

      { if (res == 3)

           /* PLACKETT-BURMAN DESIGN  */
     {
            row = 36;
      cols = row;
            orthog = 0;
            totrow = row + ceps*(reps+1);
            plackett(row,cols,ceps,totrow);
      cols = 33;}
        else  printf("DESIGN NOT AVAILABLE\n");
      }
   else if (factor == 33)
```

```
  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
        row = 36;
   cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
   cols = 34;}
    else  printf("DESIGN NOT AVAILABLE\n");
   }
else if (factor == 34)

  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
        row = 36;
   cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
   cols = 35;}
    else  printf("DESIGN NOT AVAILABLE\n");
   }
else if (factor == 35)

  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
   row = 36;
   cols = row;
        orthog = 0;
        totrow = row + ceps*(reps+1);
        plackett(row,cols,ceps,totrow);
   cols = 36;}
    else  printf("DESIGN NOT AVAILABLE\n");
   }
else if (factor == 36)

  { if (res == 3)

        /* PLACKETT-BURMAN DESIGN  */
  {
        row = 40;
   cols = row;
        orthog = 0;
```

```
              totrow = row + ceps*(reps+1);
              plackett(row,cols,ceps,totrow);
       cols = 37;}
         else  printf("DESIGN NOT AVAILABLE\n");
       }
   else if (factor == 37)

      { if (res == 3)

            /* PLACKETT-BURMAN DESIGN   */
     {
              row = 40;
       cols = row;
              orthog = 0;
              totrow = row + ceps*(reps+1);
              plackett(row,cols,ceps,totrow);
       cols = 38;}
         else  printf("DESIGN NOT AVAILABLE\n");
       }
   else if (factor == 38)

      { if (res == 3)

            /* PLACKETT-BURMAN DESIGN   */
     {
              row = 40;
       cols = row;
              orthog = 0;
              totrow = row + ceps*(reps+1);
              plackett(row,cols,ceps,totrow);
       cols = 39;}
         else  printf("DESIGN NOT AVAILABLE\n");
       }
   else if (factor == 39)

      { if (res == 3)

            /* PLACKETT-BURMAN DESIGN   */
     {
              row = 40;
       cols = row;
              orthog = 0;
              totrow = row + ceps*(reps+1);
              plackett(row,cols,ceps,totrow);
       cols = 40;}
         else  printf("DESIGN NOT AVAILABLE\n");
       }

   else printf("NO DESIGNS AVAILABLE FOR THAT NUMBER OF
FACTORS\n");
```

```
  }

 else printf("YOU MUST CHOOSE TWO FACTOR LEVELS\n");

if (factor >= 12) pb = 1;
/* IF PLACKETT-BURMAN DESIGNS THEN NO REPLICATION POSSIBLE */
if (pb == 1) {
  reps = 0;
  desfac = factor;
}
if (ceps == 0 && reps == 0) c = row;
if (ceps >= 1 && reps == 0) c = row + 1;
if (pb == 0 && ceps == 0 && reps >= 1) c = pow(2,desfac);
   else if (pb == 1 && ceps == 0 && reps >= 1) c = row;
if (pb == 0 && ceps >= 1 && reps >= 1) c = pow(2,desfac) + 1;
   else if (pb == 1 && ceps >= 1 && reps >= 1)  c = row + 1;

/*  SEND DESIGN INFORMATION TO FILE EXP.DES */
outfile = fopen("exp.des", "w");

fprintf(outfile," %2d",gpscreen);
if (gpscreen != 0)
  fprintf(outfile," %2d",gsfactor);
else
  fprintf(outfile," %2d",factor);
fprintf(outfile," %2d",totrow);
fprintf(outfile," %2d",cols);
fprintf(outfile," %2d",desfac);
fprintf(outfile," %2d",orthog);
fprintf(outfile," %2d",choice);
fprintf(outfile," %2d",level);
fprintf(outfile," %2d",ceps);
fprintf(outfile," %2d",reps);
fprintf(outfile," %2d",c);
fprintf(outfile," %2d\n",row);


for (i=1; i<=cols; i++)
  fprintf(outfile,"%4g ",desmat[0][i]);
fprintf(outfile,"\n");

for (i=1; i<=totrow; i++) {
  for (j=1; j<=cols; j++)
    fprintf(outfile,"%4g ",desmat[i][j]);
  fprintf(outfile,"\n");
}
fclose(outfile);
    printf("NUMBER OF GROUPS %d\n",gpscreen);
    if (gpscreen != 0)
```

```
      printf("NUMBER OF <GROUP SCREENING> ACTUAL FACTORS
%d\n",gsfactor);
    else
      printf("NUMBER OF ACTUAL FACTORS %d\n",factor);
    printf("NUMBER OF DESIGN FACTORS %d\n",desfac);
    printf("NUMBER OF ROWS IN CORE DESIGN %d\n",row);
    printf("TOTAL NUMBER OF ROWS %d\n",totrow);
    printf("NUMBER OF COLUMNS %3d\n",cols);
    printf("NUMBER OF CENTER POINTS %3d\n",ceps);
    printf("NUMBER OF OVERALL DESIGN REPS %3d\n",reps);
    printf("\n ****************** \n PRESS RETURN TO CONTINUE
\n");
    wait_key = getchar();

/*  PRINT OUT THE DESIGN MATRIX  */

  printf("THE DESIGN MATRIX IS\n");
  printf("\n");

   for (i=0;i<=totrow;i++)
       { for (j=1;j<=(cols);j++)
             { printf("%4.0f",desmat[i][j]); }
         printf("%3d\n",i);
       }
   printf("\n");
   if (orthog == 0) printf("NON-ORTHOGONAL DESIGN\n");
   printf("\n ****************** \n PRESS RETURN TO CONTINUE
\n");
   wait_key = getchar();
}
/*   END OF PROGRAM MAIN  */



 /* SUBROUTINE DESIGN GENERATES THE DESIGN MATRIX AND
INTERACTIONS */
int design(facs,row,cep,totrow)

{  int i,j,k,l,m,p,col;

/* INITIALIZE # FACTORS, ROWS A D COLUMNS  */

    col = facs + 1;


/* INITIATE THE DESIGN MATRIX WITH 1s  */

   for (i=0;i<=totrow;i++)
       for (j=0;j<=64;j++)
```

```
          { desmat[i][j] = 1; }

   m = 1;
   p = 2;

/* CREATE CORE MAIN EFFECTS DESIGN MATRIX */

   for (j=2;j<=col;j++)
          { l=1;
            while(l<=row)
              {  for (i=l;i<=(l-1+m);i++)
                      desmat[i][j] = -1;

                      l = l+p;
              }
            m = m*2;
            p = p*2;
          }

/* ADD CENTER POINTS, IF REQUESTED */

if (cep >= 1)
  {
    for (i=row+1; i<=totrow; i++)
        for (j=2; j<=64; j++)
             desmat[i][j] = 0;
  }

/* SUBROUTINE TO WRITE DESIGN MATRIX INTERACTIONS */
/*  new terms:                                    */

   if (facs>=2)

    { for (i=1;i<=row;i++)
          desmat[i][col+1] = desmat[i][2]*desmat[i][3];
    }

   if (facs>=3)

     { for (i=1;i<=row;i++)
         {desmat[i][col+2] = desmat[i][2]*desmat[i][4];
          desmat[i][col+3] = desmat[i][3]*desmat[i][4];
          desmat[i][col+4] =
desmat[i][2]*desmat[i][3]*desmat[i][4];
         }
     }
```

```
if (facs>=4)

 { for (i=1;i<=row;i++)
      {desmat[i][col+5] = desmat[i][2]*desmat[i][5];
       desmat[i][col+6] = desmat[i][3]*desmat[i][5];
       desmat[i][col+7] =
desmat[i][2]*desmat[i][3]*desmat[i][5];
       desmat[i][col+8] = desmat[i][4]*desmat[i][5];
       desmat[i][col+9] =
desmat[i][2]*desmat[i][4]*desmat[i][5];
       desmat[i][col+10]=
desmat[i][3]*desmat[i][4]*desmat[i][5];
       desmat[i][col+11]=
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                         desmat[i][5];

      }
  }


if (facs>=5)

 { for (i=1;i<=row;i++)
      {desmat[i][col+12] = desmat[i][2]*desmat[i][6];
       desmat[i][col+13] = desmat[i][3]*desmat[i][6];
       desmat[i][col+14] =
desmat[i][2]*desmat[i][3]*desmat[i][6];
       desmat[i][col+15] = desmat[i][4]*desmat[i][6];
       desmat[i][col+16] =
desmat[i][2]*desmat[i][4]*desmat[i][6];
       desmat[i][col+17] =
desmat[i][3]*desmat[i][4]*desmat[i][6];
       desmat[i][col+18] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                         desmat[i][6];
       desmat[i][col+19] = desmat[i][5]*desmat[i][6];
       desmat[i][col+20] =
desmat[i][2]*desmat[i][5]*desmat[i][6];
       desmat[i][col+21] =
desmat[i][3]*desmat[i][5]*desmat[i][6];
       desmat[i][col+22] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                         desmat[i][6];
       desmat[i][col+23] =
desmat[i][4]*desmat[i][5]*desmat[i][6];
       desmat[i][col+24] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                         desmat[i][6];
       desmat[i][col+25] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
```

```
                                  desmat[i][6];
          desmat[i][col+26] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                  desmat[i][5]*desmat[i][6];
          }
     }


   if (facs>=6)

     { for (i=1;i<=row;i++)
          {desmat[i][col+27] = desmat[i][2]*desmat[i][7];
           desmat[i][col+28] = desmat[i][3]*desmat[i][7];
           desmat[i][col+29] =
desmat[i][2]*desmat[i][3]*desmat[i][7];
           desmat[i][col+30] = desmat[i][4]*desmat[i][7];
           desmat[i][col+31] =
desmat[i][2]*desmat[i][4]*desmat[i][7];
           desmat[i][col+32]=
desmat[i][3]*desmat[i][4]*desmat[i][7];
           desmat[i][col+33]=
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                  desmat[i][7];

           desmat[i][col+34] = desmat[i][5]*desmat[i][7];
           desmat[i][col+35] =
desmat[i][2]*desmat[i][5]*desmat[i][7];
           desmat[i][col+36] =
desmat[i][3]*desmat[i][5]*desmat[i][7];
           desmat[i][col+37] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                                  desmat[i][7];
           desmat[i][col+38] =
desmat[i][4]*desmat[i][5]*desmat[i][7];
           desmat[i][col+39] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                                  desmat[i][7];
           desmat[i][col+40] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                                  desmat[i][7];
           desmat[i][col+41] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                  desmat[i][5]*desmat[i][7];

           desmat[i][col+42] = desmat[i][6]*desmat[i][7];
           desmat[i][col+43] =
desmat[i][2]*desmat[i][6]*desmat[i][7];
           desmat[i][col+44] =
desmat[i][3]*desmat[i][6]*desmat[i][7];
```

```
            desmat[i][col+45] =
desmat[i][2]*desmat[i][3]*desmat[i][6]*
                              desmat[i][7];
            desmat[i][col+46] =
desmat[i][4]*desmat[i][6]*desmat[i][7];
            desmat[i][col+47] =
desmat[i][2]*desmat[i][4]*desmat[i][6]*
                              desmat[i][7];
            desmat[i][col+48] =
desmat[i][3]*desmat[i][4]*desmat[i][6]*
                              desmat[i][7];
            desmat[i][col+49] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                              desmat[i][6]*desmat[i][7];

            desmat[i][col+50] =
desmat[i][5]*desmat[i][6]*desmat[i][7];
            desmat[i][col+51] =
desmat[i][2]*desmat[i][5]*desmat[i][6]*
                              desmat[i][7];
            desmat[i][col+52] =
desmat[i][3]*desmat[i][5]*desmat[i][6]*
                              desmat[i][7];
            desmat[i][col+53] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                              desmat[i][6]*desmat[i][7];
            desmat[i][col+54] =
desmat[i][4]*desmat[i][5]*desmat[i][6]*
                              desmat[i][7];
            desmat[i][col+55] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                              desmat[i][6]*desmat[i][7];
            desmat[i][col+56] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                              desmat[i][6]*desmat[i][7];
            desmat[i][col+57] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*

desmat[i][5]*desmat[i][6]*desmat[i][7];

           }
       }

}
/* END OF TWO LEVEL DESIGN */
```

```
/* FUNCTION PLACKETT GENERATES RESOLUTION III
           PLACKETT-BURMAN DESIGNS */

plackett(int row, int col, int cep, int totrow)
{
int i,j,k;

if (col == 8)
  {
    desmat[1][1] = 1;   desmat[1][2] = 1;      desmat[1][3] = 1;
    desmat[1][4] = 1;   desmat[1][5] =-1;      desmat[1][6] = 1;
    desmat[1][7] =-1;   desmat[1][8] =-1;
  }
else if (col == 12)
  {
    desmat[1][1] = 1; desmat[1][2] = 1; desmat[1][3] = 1;
    desmat[1][4] =-1; desmat[1][5] = 1; desmat[1][6] = 1;
    desmat[1][7] = 1; desmat[1][8] =-1; desmat[1][9] =-1;
    desmat[1][10] =-1;desmat[1][11] = 1;desmat[1][12] =-1;
  }
else if (col == 16)
  {
    desmat[1][1]  = 1;desmat[1][2]  = 1;desmat[1][3]  = 1;
    desmat[1][4]  = 1;desmat[1][5]  = 1;desmat[1][6]  =-1;
    desmat[1][7]  = 1;desmat[1][8]  =-1;desmat[1][9]  = 1;
    desmat[1][10] = 1;desmat[1][11] =-1;desmat[1][12] =-1;
    desmat[1][13] = 1;desmat[1][14] =-1;desmat[1][15] =-1;
    desmat[1][16] =-1;
 }
else if (col == 20)
  {
    desmat[1][1]  = 1;desmat[1][2]  = 1;desmat[1][3]  = 1;
    desmat[1][4]  =-1;desmat[1][5]  =-1;desmat[1][6]  = 1;
    desmat[1][7]  = 1;desmat[1][8]  = 1;desmat[1][9]  = 1;
    desmat[1][10] =-1;desmat[1][11] = 1;desmat[1][12] =-1;
    desmat[1][13] = 1;desmat[1][14] =-1;desmat[1][15] =-1;
    desmat[1][16] =-1;desmat[1][17] =-1;desmat[1][18] = 1;
    desmat[1][19] = 1;desmat[1][20] =-1;
  }
else if (col == 24)
  {
    desmat[1][1]  = 1;desmat[1][2]  = 1;desmat[1][3]  = 1;
    desmat[1][4]  = 1;desmat[1][5]  = 1;desmat[1][6]  = 1;
    desmat[1][7]  =-1;desmat[1][8]  = 1;desmat[1][9]  =-1;
    desmat[1][10] = 1;desmat[1][11] = 1;desmat[1][12] =-1;
    desmat[1][13] =-1;desmat[1][14] = 1;desmat[1][15] = 1;
    desmat[1][16] =-1;desmat[1][17] =-1;desmat[1][18] = 1;
    desmat[1][19] =-1;desmat[1][20] = 1;desmat[1][21] =-1;
    desmat[1][22] =-1;desmat[1][23] =-1;desmat[1][24] =-1;
  }
```

```
else if (col == 32)
  {
    desmat[1][1]  = 1;desmat[1][2]  =-1;desmat[1][3]  =-1;
    desmat[1][4]  =-1;desmat[1][5]  =-1;desmat[1][6]  = 1;
    desmat[1][7]  =-1;desmat[1][8]  = 1;desmat[1][9]  =-1;
    desmat[1][10] = 1;desmat[1][11] = 1;desmat[1][12] = 1;
    desmat[1][13] =-1;desmat[1][14] = 1;desmat[1][15] = 1;
    desmat[1][16] =-1;desmat[1][17] =-1;desmat[1][18] =-1;
    desmat[1][19] = 1;desmat[1][20] = 1;desmat[1][21] = 1;
    desmat[1][22] = 1;desmat[1][23] = 1;desmat[1][24] =-1;
    desmat[1][25] =-1;desmat[1][26] = 1;desmat[1][27] = 1;
    desmat[1][28] =-1;desmat[1][29] = 1;desmat[1][30] =-1;
    desmat[1][31] =-1;desmat[1][32] = 1;
  }
else if (col == 36)
  {
    desmat[1][1]  = 1;desmat[1][2]  =-1;desmat[1][3]  = 1;
    desmat[1][4]  =-1;desmat[1][5]  = 1;desmat[1][6]  = 1;
    desmat[1][7]  = 1;desmat[1][8]  =-1;desmat[1][9]  =-1;
    desmat[1][10] =-1;desmat[1][11] = 1;desmat[1][12] = 1;
    desmat[1][13] = 1;desmat[1][14] = 1;desmat[1][15] = 1;
    desmat[1][16] =-1;desmat[1][17] = 1;desmat[1][18] = 1;
    desmat[1][19] = 1;desmat[1][20] =-1;desmat[1][21] =-1;
    desmat[1][22] = 1;desmat[1][23] =-1;desmat[1][24] =-1;
    desmat[1][25] =-1;desmat[1][26] =-1;desmat[1][27] = 1;
    desmat[1][28] =-1;desmat[1][29] = 1;desmat[1][30] =-1;
    desmat[1][31] = 1;desmat[1][32] = 1;desmat[1][33] =-1;
    desmat[1][34] =-1;desmat[1][35] = 1;desmat[1][36] =-1;
  }
else if (col == 40)
   {desmat[1][1]  = 1;desmat[1][2]  = 1;desmat[1][3]  = 1;
    desmat[1][4]  =-1;desmat[1][5]  =-1;desmat[1][6]  = 1;
    desmat[1][7]  = 1;desmat[1][8]  = 1;desmat[1][9]  = 1;
    desmat[1][10] =-1;desmat[1][11] = 1;desmat[1][12] =-1;
    desmat[1][13] = 1;desmat[1][14] =-1;desmat[1][15] =-1;
    desmat[1][16] =-1;desmat[1][17] =-1;desmat[1][18] = 1;
    desmat[1][19] = 1;desmat[1][20] =-1;desmat[1][21] = 1;
    desmat[1][22] = 1;desmat[1][23] = 1;desmat[1][24] =-1;
    desmat[1][25] =-1;desmat[1][26] = 1;desmat[1][27] = 1;
    desmat[1][28] = 1;desmat[1][29] = 1;desmat[1][30] =-1;
    desmat[1][31] = 1;desmat[1][32] =-1;desmat[1][33] = 1;
    desmat[1][34] =-1;desmat[1][35] =-1;desmat[1][36] =-1;
    desmat[1][37] =-1;desmat[1][38] = 1;desmat[1][39] = 1;
    desmat[1][40] =-1;
   }
else printf("DESIGN UNAVAILABLE FOR THE NUMBER OF FACTORS
REQUESTED\n");

for (i=2; i<=row-1; i++)
     { for (j=2; j<=col; j++)
```

```
          { k = j-1;
                                if (k <= 1) k = col;
            desmat[i][j] = desmat[i-1][k];
          }
        }

for (i=1; i<=totrow; i++)
    desmat[i][1] = 1;

for (j=2; j<=col; j++)
    desmat[row][j] = -1;

/* ADD CENTER POINTS, IF REQUESTED */

if (cep >= 1)
  {
    for (i=row+1; i<=totrow; i++)
        for (j=2; j<=64; j++)
             desmat[i][j] = 0;
  }

/* ADD LABELS TO COLUMNS  */

for (j=1; j<=col; j++)
    desmat[0][j] = j-1;

}

/* FUNCTION lev2_label ASSIGNS LABELS TO MATRIX COLUMNS */

int lev2_label(facs)

{  int i,j,k,l,m,p,col;

col = facs + 1;

for (j=1; j<=col; j++)
    desmat[0][j] = j-1;


   if (facs>=2)   desmat[0][col+1] = 12;

   if (facs>=3)

         {desmat[0][col+2] = 13;
          desmat[0][col+3] = 23;
          desmat[0][col+4] = 123;
         }

   if (facs>=4)
```

```
        {desmat[0][col+5] = 14;
         desmat[0][col+6] = 24;
         desmat[0][col+7] = 124;
         desmat[0][col+8] = 34;
         desmat[0][col+9] = 134;
         desmat[0][col+10]= 234;
         desmat[0][col+11]= 1234;
        }

if (facs>=5)

        {desmat[0][col+12] = 15;
         desmat[0][col+13] = 25;
         desmat[0][col+14] = 125;
         desmat[0][col+15] = 35;
         desmat[0][col+16] = 135;
         desmat[0][col+17] = 235;
         desmat[0][col+18] = 1235;
         desmat[0][col+19] = 45;
         desmat[0][col+20] = 145;
         desmat[0][col+21] = 245;
         desmat[0][col+22] = 1245;
         desmat[0][col+23] = 345;
         desmat[0][col+24] = 1345;
         desmat[0][col+25] = 2345;
         desmat[0][col+26] = 12345;
        }

if (facs>=6)

        {desmat[0][col+27] = 16;
         desmat[0][col+28] = 26;
         desmat[0][col+29] = 126;
         desmat[0][col+30] = 36;
         desmat[0][col+31] = 136;
         desmat[0][col+32]=  236;
         desmat[0][col+33]=  1236;
         desmat[0][col+34] = 46;
         desmat[0][col+35] = 146;
         desmat[0][col+36] = 246;
         desmat[0][col+37] = 1246;
         desmat[0][col+38] = 346;
         desmat[0][col+39] = 1346;
         desmat[0][col+40] = 2346;
         desmat[0][col+41] = 12346;
         desmat[0][col+42] = 56;
         desmat[0][col+43] = 156;
         desmat[0][col+44] = 256;
         desmat[0][col+45] = 1256;
```

```
            desmat[0][col+46] = 356;
            desmat[0][col+47] = 1356;
            desmat[0][col+48] = 2356;
            desmat[0][col+49] = 12356;
            desmat[0][col+50] = 456;
            desmat[0][col+51] = 1456;
            desmat[0][col+52] = 2456;
            desmat[0][col+53] = 12456;
            desmat[0][col+54] = 3456;
            desmat[0][col+55] = 13456;
            desmat[0][col+56] = 23456;
            desmat[0][col+57] = 123456;
           }
}
```

LEVEL3.C

```
/* THIS PROGRAM GENERATES 3 LEVEL DESIGN MATRICES */

# include <stdio.h>
# include <math.h>

extern float alpha,desmat[65][130];
extern int gpscreen,factor,level;
extern int res,choice,orthog;
extern int ceps,reps;

/* MAIN DETERMINES # FACTORS, RESOLUTION, AND DESIGN */

cdesign3()

{
FILE *outfile;
int row,desfac,cols,totrow;
int i,j,k,l;
char wait_key;
int c;

/* GENERATE 3 LEVEL [1-0,,1] DESIGN MATRIX */

 if (level == 3)
     {

     if (factor == 2)
      {

       if (choice == 1)

          /* 3 level 2**2 full factorial */
          { desfac = 2;
            row = pow(3,desfac)*(reps+1);
            totrow = row + ceps*(reps+1);
            desthree(desfac,row,ceps,totrow,choice);
            label(desfac);
          }

       else if (choice == 2)

         /* 3 level 2**2 central composite design for 2 factors */
          { desfac = 2;
            row = pow(2,desfac)*(reps+1);
            totrow = row + (ceps+(desfac*2))*(reps+1);
            desccd(desfac,row,totrow,alpha,ceps);
            label(desfac);
          }
```

```
    else printf("DESIGN UNAVAILABLE\n");

  }


  else if (factor == 3)
   {

    if (choice == 1)

       /* 3 level 2**3 full factorial */
      { desfac = 3;
        row = pow(3,desfac)*(reps+1);
        totrow = row + ceps*(reps+1);
        desthree(desfac,row,ceps,totrow,choice);
        label(desfac);
      }

    else if (choice == 2)

      /* 3 level 2**3 central composite design for 3 factors */
       { desfac = 3;
         row = pow(2,desfac)*(reps+1);
         totrow = row + (ceps+(desfac*2))*(reps+1);
         desccd(desfac,row,totrow,alpha,ceps);
         label(desfac);
       }

     else if (choice == 3)

      /* BOX-BEHNKEN 3 FACTOR DESIGN  */
       { desfac = 3;
         row = 12;
         cols = 10;
         ceps = 3;
         totrow = 15;
         box3(desfac,totrow);
         boxint(desfac,totrow,row,choice);
         boxlabel(desfac);
       }
    }

 else if (factor == 4)
  {

   if (choice==1)

      /* 3 level 2**4 full factorial */
      { desfac = 4;
```

```
      row = pow(3,desfac)*(reps+1);
      totrow = row + ceps*(reps+1);
      box4(desfac,totrow);
      desthree(desfac,row,ceps,totrow,choice);
      label(desfac);
    }

  else if (choice == 2)

    /* 3 level 2**4 central composite design for 4 factors */
     { desfac = 4;
       row = pow(2,desfac)*(reps+1);
       totrow = row + (ceps+(desfac*2))*(reps+1);
       desccd(desfac,row,totrow,alpha,ceps);
       label(desfac);
     }

  else if (choice == 3)

    /* BOX-BEHNKEN 4 FACTOR DESIGN  */
     { desfac = 4;
       row = 24;
       cols = 15;
       ceps = 3;
       totrow = 27;
       box4(desfac,totrow);
       boxint(desfac,totrow,row,choice);
       boxlabel(desfac);
    }


 }

else if (factor == 5)
 {

  if (choice==1)

    /* 3 level 2**5 full factorial */
    { printf("3**5 DESIGN UNAVAILABLE \n");
    }


  else if (choice == 2)

    /* 3 level 2**5 central composite design for 5 factors */
     { desfac = 5;
       row = pow(2,desfac)*(reps+1);
       totrow = row + (ceps+(desfac*2))*(reps+1);
```

```
          descd(desfac,row,totrow,alpha,ceps);
          label(desfac);
        }

     else if (choice == 3)

       /* BOX-BEHNKEN 5 FACTOR DESIGN  */
        { desfac = 5;
          row = 40;
          cols = 21;
          ceps = 6;
          totrow = 46;
          box5(desfac,totrow);
          boxint(desfac,totrow,row,choice);
          boxlabel(desfac);
        }

   }

  else if (factor == 6)
   {

    if (choice==1)

       /* 3 level 2**6 full factorial */
       { printf("3**6 DESIGN UNAVAILABLE \n");
       }


    else if (choice == 2)

       /* 3 level 2**6 central composite design for 5 factors */
        { desfac = 6;
          row = pow(2,desfac)*(reps+1);
          totrow = row + (ceps+(desfac*2))*(reps+1);
          desccd(desfac,row,totrow,alpha,ceps);
          label(desfac);
        }

     else if (choice == 3)

       /* BOX-BEHNKEN 6 FACTOR DESIGN  */
        { desfac = 6;
          row = 48;
          cols = 28;
          ceps = 6;
          totrow = 54;
          box6(desfac,totrow);
          boxint(desfac,totrow,row,choice);
          boxlabel(desfac);
```

```
        }
    }
   else printf("THREE LEVEL DESIGN UNAVAILABLE >6 FACTORS\n");
   }
 else printf("YOU MUST CHOOSE THREE FACTOR LEVELS\n");

gpscreen = 0;
if (choice == 1 && ceps >= 1) orthog = 0;
if (choice == 3) orthog = 0;

if (choice == 1 || choice == 2)  cols = pow(2,desfac) + desfac;

if (choice == 1)
  {
    if (ceps == 0) c = pow(3,desfac);
    else c = pow(3,desfac) + 1;
  }
if (choice == 2) c = pow(2,desfac) + 2*desfac + 1;

if (choice == 3) c = row + 1;


/* SEND DESIGN INFORMATION TO EXP.DES  */

outfile = fopen("exp.des", "w");

fprintf(outfile," %2d",gpscreen);
fprintf(outfile," %2d",factor);
fprintf(outfile," %2d",totrow);
fprintf(outfile," %2d",cols);
fprintf(outfile," %2d",desfac);
fprintf(outfile," %2d",orthog);
fprintf(outfile," %2d",choice);
fprintf(outfile," %2d",level);
fprintf(outfile," %2d",ceps);
fprintf(outfile," %2d",reps);
fprintf(outfile," %2d",c);
fprintf(outfile," %2d\n",row);

for (j=1;j<=cols;j++)
  fprintf(outfile,"%4g ",desmat[0][j]);
fprintf(outfile,"\n");

for (i=1;i<=totrow;i++) {
  for (j=1;j<=cols;j++)
```

```
      fprintf(outfile,"%4g ",desmat[i][j]);
    fprintf(outfile,"\n");
  }
  fclose(outfile);  /*  Close the outfile for writing */

  /*  PRINT OUT DESIGN INFORMATION  */

     printf("NUMBER OF ACTUAL FACTORS %3d\n",factor);
     printf("NUMBER OF ROWS IN CORE DESIGN %3d\n",row);
     printf("TOTAL NUMBER OF ROWS %3d\n",totrow);
     printf("NUMBER OF COLUMNS %3d\n",cols);
     printf("NUMBER OF CENTER POINTS %3d\n",ceps);
     printf("NUMBER OF OVERALL DESIGN REPS %3d\n",reps);
     printf("\n ********************** \n PRESS RETURN TO CONTINUE
 \n");
     wait_key = getchar();

 /*  PRINT OUT THE DESIGN MATRIX  */

    printf("THE DESIGN MATRIX IS\n");
    printf("\n");

 if (choice==1 || choice==3)
   {
    for (i=0;i<=totrow;i++)
        { for (j=1;j<=(cols);j++)
              { printf("%4.0f",desmat[i][j]); }
          printf("%3d\n",i);
        }
    printf("\n");
   }

 if (choice == 2)
   {
    for (i=0;i<=totrow;i++)
        { for (j=1;j<=(cols);j++)
              { printf("%5.2f",desmat[i][j]); }
          printf("%3d\n",i);
        }
    printf("\n");
   }
 printf("\n ********************** \n PRESS RETURN TO CONTINUE
 \n");
 wait_key = getchar();

 }
 /*   END OF PROGRAM MAIN  */
```

```
 /* SUBROUTINE DESCCD GENERATES THE CCD DESIGN MATRIX AND
INTERACTIONS */
desccd(facs, row, totrow, alph, cep)
int facs, row, totrow, cep;
float alph;

{
   int i,j,k,l,m,p,col;
   float diff;

/* INITIALIZE # FACTORS, ROWS AND COLUMNS   */

    col = facs + 1;

/* INITIATE THE DESIGN MATRIX WITH 1s   */

   for (i=0;i<=totrow;i++)
       for (j=0;j<=64;j++)
           { desmat[i][j] = 1; }

   m = 1;
   p = 2;

/* CREATE CORE MAIN EFFECTS DESIGN MATRIX */

   for (j=2;j<=col;j++)
          { l=1;
            while(l<=row)
              {  for (i=l;i<=(l-1+m);i++)
                      desmat[i][j] = -1;

                      l = l+p;
              }
            m = m*2;
            p = p*2;
          }

/* CREATE THE EXTRA CCD ROWS   */

   /*  PUT ZEROs IN ROWS FOR CCD DESIGNS */

          for (k=(row+1);k<=totrow;k++)
                                 {    for (j=2;j<=pow(2,col);j++)
                                                desmat[k][j] = 0;
            }

   /*  PUT ALPHAs IN ROWS FOR CCD DESIGNS   */

          l = row+cep; /* LAST CENTER POINT ROW */
          p = 1;
```

```
      while (l < totrow)

       {k = 2;

        for (m=l+1;m<=l+(2*facs);m=m+2)

            { desmat[m][k]   = -(alph);
              desmat[m+1][k] =   alph;

           k = k +1;
          }

          l = row + p*cep + p*(2*facs) + cep;
  /*  INCREMENT WITH:  CENTER PTS    ALPHA ROWS     */
          p = p+1;
         }


/* SUBROUTINE TO WRITE DESIGN MATRIX INTERACTIONS */

/*  new terms:                                    */

    if (facs == 2)

         /*  ADD INTERACTION TERMS FOR CCD DESIGNS */

       { for (i=1;i<=totrow;i++)
         {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
          desmat[i][col+2] = desmat[i][3]*desmat[i][3];
          desmat[i][col+3] = desmat[i][2]*desmat[i][3];
         }
         for (i=1; i<=totrow; i++)
             for (j=col+1; j<=col+2; j++)
                { diff = (row + 2.*alph*alph)/totrow;
                  desmat[i][j] = desmat[i][j] - diff;
                }
       }

    else if (facs == 3)

          /* ADD SQUARED TERMS FOR CCD DESIGNS */

       { for (i=1;i<=totrow;i++)
         {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
          desmat[i][col+2] = desmat[i][3]*desmat[i][3];
          desmat[i][col+3] = desmat[i][4]*desmat[i][4];
          desmat[i][col+4] = desmat[i][2]*desmat[i][3];
          desmat[i][col+5] = desmat[i][2]*desmat[i][4];
          desmat[i][col+6] = desmat[i][3]*desmat[i][4];
```

```
         desmat[i][col+7] =
desmat[i][2]*desmat[i][3]*desmat[i][4];
        }
        for (i=1; i<=totrow; i++)
            for (j=col+1; j<=col+3; j++)
               { diff = (row + 2.*alph*alph)/totrow;
                 desmat[i][j] = desmat[i][j] - diff;
               }
      }

    else if (facs == 4)

         /* ADD SQUARED TERMS FOR CCD DESIGNS */

      { for (i=1;i<=totrow;i++)
        {desmat[i][col+1]  = desmat[i][2]*desmat[i][2];
         desmat[i][col+2]  = desmat[i][3]*desmat[i][3];
         desmat[i][col+3]  = desmat[i][4]*desmat[i][4];
         desmat[i][col+4]  = desmat[i][5]*desmat[i][5];
         desmat[i][col+5]  = desmat[i][2]*desmat[i][3];
         desmat[i][col+6]  = desmat[i][2]*desmat[i][4];
         desmat[i][col+7]  = desmat[i][3]*desmat[i][4];
         desmat[i][col+8]  =
desmat[i][2]*desmat[i][3]*desmat[i][4];
         desmat[i][col+9]  = desmat[i][2]*desmat[i][5];
         desmat[i][col+10] = desmat[i][3]*desmat[i][5];
         desmat[i][col+11] =
desmat[i][2]*desmat[i][3]*desmat[i][5];
         desmat[i][col+12] = desmat[i][4]*desmat[i][5];
         desmat[i][col+13] =
desmat[i][2]*desmat[i][4]*desmat[i][5];
         desmat[i][col+14] =
desmat[i][3]*desmat[i][4]*desmat[i][5];
         desmat[i][col+15] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                             desmat[i][5];
        }
        for (i=1; i<=totrow; i++)
            for (j=col+1; j<=col+4; j++)
               { diff = (row + 2.*alph*alph)/totrow;
                 desmat[i][j] = desmat[i][j] - diff;
               }
      }

    else if (facs == 5)

      { for (i=1;i<=totrow;i++)
        {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
         desmat[i][col+2] = desmat[i][3]*desmat[i][3];
         desmat[i][col+3] = desmat[i][4]*desmat[i][4];
```

```
            desmat[i][col+4] = desmat[i][5]*desmat[i][5];
            desmat[i][col+5] = desmat[i][6]*desmat[i][6];
            desmat[i][col+6] = desmat[i][2]*desmat[i][3];
            desmat[i][col+7] = desmat[i][2]*desmat[i][4];
            desmat[i][col+8] = desmat[i][3]*desmat[i][4];
            desmat[i][col+9] =
desmat[i][2]*desmat[i][3]*desmat[i][4];
            desmat[i][col+10] = desmat[i][2]*desmat[i][5];
            desmat[i][col+11] = desmat[i][3]*desmat[i][5];
            desmat[i][col+12] =
desmat[i][2]*desmat[i][3]*desmat[i][5];
            desmat[i][col+13] = desmat[i][4]*desmat[i][5];
            desmat[i][col+14] =
desmat[i][2]*desmat[i][4]*desmat[i][5];
            desmat[i][col+15] =
desmat[i][3]*desmat[i][4]*desmat[i][5];
            desmat[i][col+16] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                    desmat[i][5];
            desmat[i][col+17] = desmat[i][2]*desmat[i][6];
            desmat[i][col+18] = desmat[i][3]*desmat[i][6];
            desmat[i][col+10] =
desmat[i][2]*desmat[i][3]*desmat[i][6];
            desmat[i][col+20] = desmat[i][4]*desmat[i][6];
            desmat[i][col+21] =
desmat[i][2]*desmat[i][4]*desmat[i][6];
            desmat[i][col+22] =
desmat[i][3]*desmat[i][4]*desmat[i][6];
            desmat[i][col+23] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                    desmat[i][6];
            desmat[i][col+24] = desmat[i][5]*desmat[i][6];
            desmat[i][col+25] =
desmat[i][2]*desmat[i][5]*desmat[i][6];
            desmat[i][col+26] =
desmat[i][3]*desmat[i][5]*desmat[i][6];
            desmat[i][col+27] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                                    desmat[i][6];
            desmat[i][col+28] =
desmat[i][4]*desmat[i][5]*desmat[i][6];
            desmat[i][col+29] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                                    desmat[i][6];
            desmat[i][col+30] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                                    desmat[i][6];
            desmat[i][col+31] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                    desmat[i][5]*desmat[i][6];
```

```
        }
        for (i=1; i<=totrow; i++)
            for (j=col+1; j<=col+5; j++)
               { diff = (row + 2.*alph*alph)/totrow;
                 desmat[i][j] = desmat[i][j] - diff;
               }
     }

   else if (facs == 6)

     { for (i=1;i<=totrow;i++)
        {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
         desmat[i][col+2] = desmat[i][3]*desmat[i][3];
         desmat[i][col+3] = desmat[i][4]*desmat[i][4];
         desmat[i][col+4] = desmat[i][5]*desmat[i][5];
         desmat[i][col+5] = desmat[i][6]*desmat[i][6];
         desmat[i][col+6] = desmat[i][7]*desmat[i][7];
         desmat[i][col+7] = desmat[i][2]*desmat[i][3];
         desmat[i][col+8] = desmat[i][2]*desmat[i][4];
         desmat[i][col+9] = desmat[i][3]*desmat[i][4];
         desmat[i][col+10]=
desmat[i][2]*desmat[i][3]*desmat[i][4];
         desmat[i][col+11] = desmat[i][2]*desmat[i][5];
         desmat[i][col+12] = desmat[i][3]*desmat[i][5];
         desmat[i][col+13] =
desmat[i][2]*desmat[i][3]*desmat[i][5];
         desmat[i][col+14] = desmat[i][4]*desmat[i][5];
         desmat[i][col+15] =
desmat[i][2]*desmat[i][4]*desmat[i][5];
         desmat[i][col+16] =
desmat[i][3]*desmat[i][4]*desmat[i][5];
         desmat[i][col+17] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                             desmat[i][5];
         desmat[i][col+18] = desmat[i][2]*desmat[i][6];
         desmat[i][col+19] = desmat[i][3]*desmat[i][6];
         desmat[i][col+20] =
desmat[i][2]*desmat[i][3]*desmat[i][6];
         desmat[i][col+21] = desmat[i][4]*desmat[i][6];
         desmat[i][col+22] =
desmat[i][2]*desmat[i][4]*desmat[i][6];
         desmat[i][col+23] =
desmat[i][3]*desmat[i][4]*desmat[i][6];
         desmat[i][col+24] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                              desmat[i][6];
         desmat[i][col+25] = desmat[i][5]*desmat[i][6];
         desmat[i][col+26] =
desmat[i][2]*desmat[i][5]*desmat[i][6];
```

```
        desmat[i][col+27] =
desmat[i][3]*desmat[i][5]*desmat[i][6];
        desmat[i][col+28] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                        desmat[i][6];
        desmat[i][col+29] =
desmat[i][4]*desmat[i][5]*desmat[i][6];
        desmat[i][col+30] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                        desmat[i][6];
        desmat[i][col+31] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                        desmat[i][6];
        desmat[i][col+32] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                        desmat[i][5]*desmat[i][6];
        desmat[i][col+33] = desmat[i][2]*desmat[i][7];
        desmat[i][col+34] = desmat[i][3]*desmat[i][7];
        desmat[i][col+35] =
desmat[i][2]*desmat[i][3]*desmat[i][7];
        desmat[i][col+36] = desmat[i][4]*desmat[i][7];
        desmat[i][col+37] =
desmat[i][2]*desmat[i][4]*desmat[i][7];
        desmat[i][col+38]=
desmat[i][3]*desmat[i][4]*desmat[i][7];
        desmat[i][col+39]=
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                        desmat[i][7];

        desmat[i][col+40] = desmat[i][5]*desmat[i][7];
        desmat[i][col+41] =
desmat[i][2]*desmat[i][5]*desmat[i][7];
        desmat[i][col+42] =
desmat[i][3]*desmat[i][5]*desmat[i][7];
        desmat[i][col+43] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                        desmat[i][7];
        desmat[i][col+44] =
desmat[i][4]*desmat[i][5]*desmat[i][7];
        desmat[i][col+45] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                        desmat[i][7];
        desmat[i][col+46] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                        desmat[i][7];
        desmat[i][col+47] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                        desmat[i][5]*desmat[i][7];

        desmat[i][col+48] = desmat[i][6]*desmat[i][7];
```

```
            desmat[i][col+49] =
desmat[i][2]*desmat[i][6]*desmat[i][7];
            desmat[i][col+50] =
desmat[i][3]*desmat[i][6]*desmat[i][7];
            desmat[i][col+51] =
desmat[i][2]*desmat[i][3]*desmat[i][6]*
                                        desmat[i][7];
            desmat[i][col+52] =
desmat[i][4]*desmat[i][6]*desmat[i][7];
            desmat[i][col+53] =
desmat[i][2]*desmat[i][4]*desmat[i][6]*
                                        desmat[i][7];
            desmat[i][col+54] =
desmat[i][3]*desmat[i][4]*desmat[i][6]*
                                        desmat[i][7];
            desmat[i][col+55] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                        desmat[i][6]*desmat[i][7];

            desmat[i][col+56] =
desmat[i][5]*desmat[i][6]*desmat[i][7];
            desmat[i][col+57] =
desmat[i][2]*desmat[i][5]*desmat[i][6]*
                                        desmat[i][7];
            desmat[i][col+58] =
desmat[i][3]*desmat[i][5]*desmat[i][6]*
                                        desmat[i][7];
            desmat[i][col+59] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                                        desmat[i][6]*desmat[i][7];
            desmat[i][col+60] =
desmat[i][4]*desmat[i][5]*desmat[i][6]*
                                        desmat[i][7];
            desmat[i][col+61] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                                        desmat[i][6]*desmat[i][7];
            desmat[i][col+62] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                                        desmat[i][6]*desmat[i][7];
            desmat[i][col+63] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
 desmat[i][5]*desmat[i][6]*desmat[i][7];
          }
           for (i=1; i<=totrow; i++)
               for (j=col+1; j<=col+6; j++)
                  { diff = (row + 2.*alph*alph)/totrow;
                    desmat[i][j] = desmat[i][j] - diff;
                  }
      }
}
```

```
/* SUBROUTINE DESTHREE GENERATES THREE LEVEL DESIGNS */

int desthree(facs,row,cep,totrow,choice)

{
   int i,j,k,l,m,p,col;

/* INITIALIZE # FACTORS, ROWS, AND COLUMNS  */

    col = facs + 1;

/* INITIATE THE DESIGN MATRIX WITH 1s  */

if (choice == 1)
 {

   for (i=0;i<=totrow;i++)
       for (j=0;j<=64;j++)
           { desmat[i][j] = 1; }

/* CREATE FULL THREE LEVEL CORE MAIN EFFECTS DESIGN MATRIX */

   m = 1;
   p = 3;

   for (j=2;j<=col;j++)
       { l=1;
         while(l<=row)
             {  for (i=l;i<=(l-1+m);i++)
                    desmat[i][j] = -1;
                    l = l+p;
             }
        m = m*3;
        p = p*3;
      }

   m = 1;
   p = 3;
   k = 0;

   for (j=2;j<=col;j++)
       { l=pow(3,k)+1;
         while(l<=row)
            {  for (i=l;i<=(l-1+m);i++)
                    desmat[i][j] = 0;
                    l = l+p;
            }
         m = m*3;
         p = p*3;
```

```
          k = k+1;
      }


/* ADD CENTER POINTS, IF REQUESTED */

if (cep >= 1)
  {
    for (i=row+1; i<=totrow; i++)
        for (j=2; j<=64; j++)
             desmat[i][j] = 0;
  }


 }

/* SUBROUTINE TO WRITE DESIGN MATRIX INTERACTIONS */

/*  new terms:                                     */

   if (facs==2)

     { for (i=1;i<=row;i++)
        { desmat[i][col+1] = desmat[i][2]*desmat[i][2];
          desmat[i][col+2] = desmat[i][3]*desmat[i][3];
          desmat[i][col+3] = desmat[i][2]*desmat[i][3];
        }
       if (choice == 1)
       {for (i=1; i<=totrow; i++)
           for (j=col+1; j<=col+2; j++)
               desmat[i][j] = desmat[i][j] - (2./3.*row)/totrow;
       }
     }

   if (facs==3)

     { for (i=1;i<=row;i++)
         {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
          desmat[i][col+2] = desmat[i][3]*desmat[i][3];
          desmat[i][col+3] = desmat[i][4]*desmat[i][4];
          desmat[i][col+4] = desmat[i][2]*desmat[i][3];
          desmat[i][col+5] = desmat[i][2]*desmat[i][4];
          desmat[i][col+6] = desmat[i][3]*desmat[i][4];
          desmat[i][col+7] =
desmat[i][2]*desmat[i][3]*desmat[i][4];
         }
       if (choice == 1)
       {for (i=1; i<=totrow; i++)
           for (j=col+1; j<=col+3; j++)
```

```
                desmat[i][j] = desmat[i][j] - (2./3.*row)/totrow;
              }
      }

   if (facs==4)

    { for (i=1;i<=row;i++)
          {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
           desmat[i][col+2] = desmat[i][3]*desmat[i][3];
           desmat[i][col+3] = desmat[i][4]*desmat[i][4];
           desmat[i][col+4] = desmat[i][5]*desmat[i][5];
           desmat[i][col+5] = desmat[i][2]*desmat[i][3];
           desmat[i][col+6] = desmat[i][2]*desmat[i][4];
           desmat[i][col+7] = desmat[i][3]*desmat[i][4];
           desmat[i][col+8] =
desmat[i][2]*desmat[i][3]*desmat[i][4];
           desmat[i][col+9] = desmat[i][2]*desmat[i][5];
           desmat[i][col+10]= desmat[i][3]*desmat[i][5];
           desmat[i][col+11]=
desmat[i][2]*desmat[i][3]*desmat[i][5];
           desmat[i][col+12]= desmat[i][4]*desmat[i][5];
           desmat[i][col+13]=
desmat[i][2]*desmat[i][4]*desmat[i][5];
           desmat[i][col+14]=
desmat[i][3]*desmat[i][4]*desmat[i][5];
           desmat[i][col+15]=
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                     desmat[i][5];
          }
        if (choice == 1)
        {for (i=1; i<=totrow; i++)
            for (j=col+1; j<=col+4; j++)
                desmat[i][j] = desmat[i][j] - (2./3.*row)/totrow;
        }
         }

   if (facs==5)

    { for (i=1;i<=row;i++)
          {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
           desmat[i][col+2] = desmat[i][3]*desmat[i][3];
           desmat[i][col+3] = desmat[i][4]*desmat[i][4];
           desmat[i][col+4] = desmat[i][5]*desmat[i][5];
           desmat[i][col+5] = desmat[i][6]*desmat[i][6];
           desmat[i][col+6] = desmat[i][2]*desmat[i][3];
           desmat[i][col+7] = desmat[i][2]*desmat[i][4];
           desmat[i][col+8] = desmat[i][3]*desmat[i][4];
           desmat[i][col+9] =
desmat[i][2]*desmat[i][3]*desmat[i][4];
           desmat[i][col+10]= desmat[i][2]*desmat[i][5];
```

```
            desmat[i][col+11]= desmat[i][3]*desmat[i][5];
            desmat[i][col+12]=
desmat[i][2]*desmat[i][3]*desmat[i][5];
            desmat[i][col+13]= desmat[i][4]*desmat[i][5];
            desmat[i][col+14]=
desmat[i][2]*desmat[i][4]*desmat[i][5];
            desmat[i][col+15]=
desmat[i][3]*desmat[i][4]*desmat[i][5];
            desmat[i][col+16]=
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                  desmat[i][5];
            desmat[i][col+17] = desmat[i][2]*desmat[i][6];
            desmat[i][col+18] = desmat[i][3]*desmat[i][6];
            desmat[i][col+19] =
desmat[i][2]*desmat[i][3]*desmat[i][6];
            desmat[i][col+20] = desmat[i][4]*desmat[i][6];
            desmat[i][col+21] =
desmat[i][2]*desmat[i][4]*desmat[i][6];
            desmat[i][col+22] =
desmat[i][3]*desmat[i][4]*desmat[i][6];
            desmat[i][col+23] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                   desmat[i][6];
            desmat[i][col+24] = desmat[i][5]*desmat[i][6];
            desmat[i][col+25] =
desmat[i][2]*desmat[i][5]*desmat[i][6];
            desmat[i][col+26] =
desmat[i][3]*desmat[i][5]*desmat[i][6];
            desmat[i][col+27] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                                   desmat[i][6];
            desmat[i][col+28] =
desmat[i][4]*desmat[i][5]*desmat[i][6];
            desmat[i][col+29] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                                   desmat[i][6];
            desmat[i][col+30] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                                   desmat[i][6];
            desmat[i][col+31] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                   desmat[i][5]*desmat[i][6];
          }
        if (choice == 1)
        {for (i=1; i<=totrow; i++)
            for (j=col+1; j<=col+5; j++)
                desmat[i][j] = desmat[i][j] - (2./3.*row)/totrow;
        }
    }
```

```
if (facs==6)

 { for (i=1;i<=row;i++)
      {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
       desmat[i][col+2] = desmat[i][3]*desmat[i][3];
       desmat[i][col+3] = desmat[i][4]*desmat[i][4];
       desmat[i][col+4] = desmat[i][5]*desmat[i][5];
       desmat[i][col+5] = desmat[i][6]*desmat[i][6];
       desmat[i][col+6] = desmat[i][7]*desmat[i][7];
       desmat[i][col+7] = desmat[i][2]*desmat[i][3];
       desmat[i][col+8] = desmat[i][2]*desmat[i][4];
       desmat[i][col+9] = desmat[i][3]*desmat[i][4];
       desmat[i][col+10]=
desmat[i][2]*desmat[i][3]*desmat[i][4];
       desmat[i][col+11]= desmat[i][2]*desmat[i][5];
       desmat[i][col+12]= desmat[i][3]*desmat[i][5];
       desmat[i][col+13]=
desmat[i][2]*desmat[i][3]*desmat[i][5];
       desmat[i][col+14]= desmat[i][4]*desmat[i][5];
       desmat[i][col+15]=
desmat[i][2]*desmat[i][4]*desmat[i][5];
       desmat[i][col+16]=
desmat[i][3]*desmat[i][4]*desmat[i][5];
       desmat[i][col+17]=
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                              desmat[i][5];
       desmat[i][col+18] = desmat[i][2]*desmat[i][6];
       desmat[i][col+19] = desmat[i][3]*desmat[i][6];
       desmat[i][col+20] =
desmat[i][2]*desmat[i][3]*desmat[i][6];
       desmat[i][col+21] = desmat[i][4]*desmat[i][6];
       desmat[i][col+22] =
desmat[i][2]*desmat[i][4]*desmat[i][6];
       desmat[i][col+23] =
desmat[i][3]*desmat[i][4]*desmat[i][6];
       desmat[i][col+24] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                               desmat[i][6];
       desmat[i][col+25] = desmat[i][5]*desmat[i][6];
       desmat[i][col+26] =
desmat[i][2]*desmat[i][5]*desmat[i][6];
       desmat[i][col+27] =
desmat[i][3]*desmat[i][5]*desmat[i][6];
       desmat[i][col+28] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                               desmat[i][6];
       desmat[i][col+29] =
desmat[i][4]*desmat[i][5]*desmat[i][6];
       desmat[i][col+30] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
```

```
                                desmat[i][6];
        desmat[i][col+31] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                                desmat[i][6];
        desmat[i][col+32] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                desmat[i][5]*desmat[i][6];
        desmat[i][col+33] = desmat[i][2]*desmat[i][7];
        desmat[i][col+34] = desmat[i][3]*desmat[i][7];
        desmat[i][col+35] =
desmat[i][2]*desmat[i][3]*desmat[i][7];
        desmat[i][col+36] = desmat[i][4]*desmat[i][7];
        desmat[i][col+37] =
desmat[i][2]*desmat[i][4]*desmat[i][7];
        desmat[i][col+38]=
desmat[i][3]*desmat[i][4]*desmat[i][7];
        desmat[i][col+39]=
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                desmat[i][7];

        desmat[i][col+40] = desmat[i][5]*desmat[i][7];
        desmat[i][col+41] =
desmat[i][2]*desmat[i][5]*desmat[i][7];
        desmat[i][col+42] =
desmat[i][3]*desmat[i][5]*desmat[i][7];
        desmat[i][col+43] =
desmat[i][2]*desmat[i][3]*desmat[i][5]*
                                desmat[i][7];
        desmat[i][col+44] =
desmat[i][4]*desmat[i][5]*desmat[i][7];
        desmat[i][col+45] =
desmat[i][2]*desmat[i][4]*desmat[i][5]*
                                desmat[i][7];
        desmat[i][col+46] =
desmat[i][3]*desmat[i][4]*desmat[i][5]*
                                desmat[i][7];
        desmat[i][col+47] =
desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                desmat[i][5]*desmat[i][7];

        desmat[i][col+48] = desmat[i][6]*desmat[i][7];
        desmat[i][col+49] =
desmat[i][2]*desmat[i][6]*desmat[i][7];
        desmat[i][col+50] =
desmat[i][3]*desmat[i][6]*desmat[i][7];
        desmat[i][col+51] =
desmat[i][2]*desmat[i][3]*desmat[i][6]*
                                desmat[i][7];
        desmat[i][col+52] =
desmat[i][4]*desmat[i][6]*desmat[i][7];
```

```
            desmat[i][col+53] =
      desmat[i][2]*desmat[i][4]*desmat[i][6]*
                                    desmat[i][7];
            desmat[i][col+54] =
      desmat[i][3]*desmat[i][4]*desmat[i][6]*
                                    desmat[i][7];
            desmat[i][col+55] =
      desmat[i][2]*desmat[i][3]*desmat[i][4]*
                                    desmat[i][6]*desmat[i][7];

            desmat[i][col+56] =
      desmat[i][5]*desmat[i][6]*desmat[i][7];
            desmat[i][col+57] =
      desmat[i][2]*desmat[i][5]*desmat[i][6]*
                                    desmat[i][7];
            desmat[i][col+58] =
      desmat[i][3]*desmat[i][5]*desmat[i][6]*
                                    desmat[i][7];
            desmat[i][col+59] =
      desmat[i][2]*desmat[i][3]*desmat[i][5]*
                                    desmat[i][6]*desmat[i][7];
            desmat[i][col+60] =
      desmat[i][4]*desmat[i][5]*desmat[i][6]*
                                    desmat[i][7];
            desmat[i][col+61] =
      desmat[i][2]*desmat[i][4]*desmat[i][5]*
                                    desmat[i][6]*desmat[i][7];
            desmat[i][col+62] =
      desmat[i][3]*desmat[i][4]*desmat[i][5]*
                                    desmat[i][6]*desmat[i][7];
            desmat[i][col+63] =
      desmat[i][2]*desmat[i][3]*desmat[i][4]*

      desmat[i][5]*desmat[i][6]*desmat[i][7];
              }
          if (choice == 1)
          {for (i=1; i<=totrow; i++)
              for (j=col+1; j<=col+6; j++)
                  desmat[i][j] = desmat[i][j] - (2./3.*row)/totrow;
          }
        }
    }

/* CREATE BOX-BEHNKEN INTERACTIONS */
int boxint(facs,totrow,row,choice)
{
int i, j, col;


/* ADD CENTER POINTS */
```

```
    for (i=row+1; i<=totrow; i++)
        for (j=2; j<=64; j++)
            desmat[i][j] = 0;


/* ADD DESIGN MATRIX INTERACTIONS */

/*  new terms:                                                  */

   col = facs + 1;

   if (facs==3)

     { for (i=1;i<=row;i++)
          {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
           desmat[i][col+2] = desmat[i][3]*desmat[i][3];
           desmat[i][col+3] = desmat[i][4]*desmat[i][4];
           desmat[i][col+4] = desmat[i][2]*desmat[i][3];
           desmat[i][col+5] = desmat[i][2]*desmat[i][4];
           desmat[i][col+6] = desmat[i][3]*desmat[i][4];
          }
       for (i=1; i<=totrow; i++)
           for (j=col+1; j<=col+3; j++)
               desmat[i][j] = desmat[i][j] - (8./15.);
          }

   if (facs==4)

    { for (i=1;i<=row;i++)
          {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
           desmat[i][col+2] = desmat[i][3]*desmat[i][3];
           desmat[i][col+3] = desmat[i][4]*desmat[i][4];
           desmat[i][col+4] = desmat[i][5]*desmat[i][5];
           desmat[i][col+5] = desmat[i][2]*desmat[i][3];
           desmat[i][col+6] = desmat[i][2]*desmat[i][4];
           desmat[i][col+7] = desmat[i][3]*desmat[i][4];
           desmat[i][col+8] = desmat[i][2]*desmat[i][5];
           desmat[i][col+9] = desmat[i][3]*desmat[i][5];
           desmat[i][col+10]= desmat[i][4]*desmat[i][5];
          }
       for (i=1; i<=totrow; i++)
           for (j=col+1; j<=col+4; j++)
               desmat[i][j] = desmat[i][j] - (12./27.);
       }

   if (facs==5)

    { for (i=1;i<=row;i++)
          {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
```

```
            desmat[i][col+2] = desmat[i][3]*desmat[i][3];
            desmat[i][col+3] = desmat[i][4]*desmat[i][4];
            desmat[i][col+4] = desmat[i][5]*desmat[i][5];
            desmat[i][col+5] = desmat[i][6]*desmat[i][6];
            desmat[i][col+6] = desmat[i][2]*desmat[i][3];
            desmat[i][col+7] = desmat[i][2]*desmat[i][4];
            desmat[i][col+8] = desmat[i][3]*desmat[i][4];
            desmat[i][col+9] = desmat[i][2]*desmat[i][5];
            desmat[i][col+10]= desmat[i][3]*desmat[i][5];
            desmat[i][col+11]= desmat[i][4]*desmat[i][5];
            desmat[i][col+12]= desmat[i][2]*desmat[i][6];
            desmat[i][col+13]= desmat[i][3]*desmat[i][6];
            desmat[i][col+14]= desmat[i][4]*desmat[i][6];
            desmat[i][col+15]= desmat[i][5]*desmat[i][6];
          }
        if (choice == 1)
        {for (i=1; i<=totrow; i++)
            for (j=col+1; j<=col+5; j++)
                desmat[i][j] = desmat[i][j] - (2./3.*row)/totrow;
        }
    }

  if (facs==6)

    { for (i=1;i<=row;i++)
          {desmat[i][col+1] = desmat[i][2]*desmat[i][2];
            desmat[i][col+2] = desmat[i][3]*desmat[i][3];
            desmat[i][col+3] = desmat[i][4]*desmat[i][4];
            desmat[i][col+4] = desmat[i][5]*desmat[i][5];
            desmat[i][col+5] = desmat[i][6]*desmat[i][6];
            desmat[i][col+6] = desmat[i][7]*desmat[i][7];
            desmat[i][col+7] = desmat[i][2]*desmat[i][3];
            desmat[i][col+8] = desmat[i][2]*desmat[i][4];
            desmat[i][col+9] = desmat[i][3]*desmat[i][4];
            desmat[i][col+10]= desmat[i][2]*desmat[i][5];
            desmat[i][col+11]= desmat[i][3]*desmat[i][5];
            desmat[i][col+12]= desmat[i][4]*desmat[i][5];
            desmat[i][col+13]= desmat[i][2]*desmat[i][6];
            desmat[i][col+14]= desmat[i][3]*desmat[i][6];
            desmat[i][col+15]= desmat[i][4]*desmat[i][6];
            desmat[i][col+16]= desmat[i][5]*desmat[i][6];
            desmat[i][col+17]= desmat[i][2]*desmat[i][7];
            desmat[i][col+18]= desmat[i][3]*desmat[i][7];
            desmat[i][col+19]= desmat[i][4]*desmat[i][7];
            desmat[i][col+20]= desmat[i][5]*desmat[i][7];
            desmat[i][col+21]= desmat[i][6]*desmat[i][7];
          }
        if (choice == 1)
        {for (i=1; i<=totrow; i++)
            for (j=col+1; j<=col+6; j++)
```

```
                desmat[i][j] = desmat[i][j] - (2./3.*row)/totrow;
        }
    }
}

/* CREATE BOX-BEHNKEN CORE DESIGN */
int box3(facs,totrow)
{
int i, j, col;

    for (i=1; i<=totrow; i++)
        desmat[i][1] = 1;

    if (facs == 3)
      { desmat[1][2] =-1; desmat[1][3] =-1; desmat[1][4]  = 0;
        desmat[2][2] = 1; desmat[2][3] =-1; desmat[2][4]  = 0;
        desmat[3][2] =-1; desmat[3][3] = 1; desmat[3][4]  = 0;
        desmat[4][2] = 1; desmat[4][3] = 1; desmat[4][4]  = 0;
        desmat[5][2] =-1; desmat[5][3] = 0; desmat[5][4]  =-1;
        desmat[6][2] = 1; desmat[6][3] = 0; desmat[6][4]  =-1;
        desmat[7][2] =-1; desmat[7][3] = 0; desmat[7][4]  = 1;
        desmat[8][2] = 1; desmat[8][3] = 0; desmat[8][4]  = 1;
        desmat[9][2] = 0; desmat[9][3] =-1; desmat[9][4]  =-1;
        desmat[10][2]= 0; desmat[10][3]= 1; desmat[10][4]=-1;
        desmat[11][2]= 0; desmat[11][3]=-1; desmat[11][4]= 1;
        desmat[12][2]= 0; desmat[12][3]= 1; desmat[12][4]= 1;
        desmat[13][2]= 0; desmat[13][3]= 0; desmat[13][4]= 0;
        desmat[14][2]= 0; desmat[14][3]= 0; desmat[14][4]= 0;
        desmat[15][2]= 0; desmat[15][3]= 0; desmat[15][4]= 0;
      }
}
/* CREATE BOX-BEHNKEN CORE DESIGN */
int box4(facs,totrow)
{
int i, j, col;

    for (i=1; i<=totrow; i++)
        desmat[i][1] = 1;

    if (facs == 4)
      { desmat[1][2] =-1; desmat[1][3] =-1; desmat[1][4]  = 0;
        desmat[2][2] = 1; desmat[2][3] =-1; desmat[2][4]  = 0;
        desmat[3][2] =-1; desmat[3][3] = 1; desmat[3][4]  = 0;
        desmat[4][2] = 1; desmat[4][3] = 1; desmat[4][4]  = 0;
        desmat[5][2] = 0; desmat[5][3] = 0; desmat[5][4]  =-1;
        desmat[6][2] = 0; desmat[6][3] = 0; desmat[6][4]  = 1;
        desmat[7][2] = 0; desmat[7][3] = 0; desmat[7][4]  =-1;
        desmat[8][2] = 0; desmat[8][3] = 0; desmat[8][4]  = 1;
        desmat[9][2] =-1; desmat[9][3] = 0; desmat[9][4]  = 0;
```

```
        desmat[10][2]= 1; desmat[10][3]= 0; desmat[10][4]= 0;
        desmat[11][2]=-1; desmat[11][3]= 0; desmat[11][4]= 0;
        desmat[12][2]= 1; desmat[12][3]= 0; desmat[12][4]= 0;
        desmat[13][2]= 0; desmat[13][3]=-1; desmat[13][4]=-1;
        desmat[14][2]= 0; desmat[14][3]= 1; desmat[14][4]=-1;
        desmat[15][2]= 0; desmat[15][3]=-1; desmat[15][4]= 1;
        desmat[16][2]= 0; desmat[16][3]= 1; desmat[16][4]= 1;
        desmat[17][2]=-1; desmat[17][3]= 0; desmat[17][4]=-1;
        desmat[18][2]= 1; desmat[18][3]= 0; desmat[18][4]=-1;
        desmat[19][2]=-1; desmat[19][3]= 0; desmat[19][4]= 1;
        desmat[20][2]= 1; desmat[20][3]= 0; desmat[20][4]= 1;
        desmat[21][2]= 0; desmat[21][3]=-1; desmat[21][4]= 0;
        desmat[22][2]= 0; desmat[22][3]= 1; desmat[22][4]= 0;
        desmat[23][2]= 0; desmat[23][3]=-1; desmat[23][4]= 0;
        desmat[24][2]= 0; desmat[24][3]= 1; desmat[24][4]= 0;
        desmat[25][2]= 0; desmat[25][3]= 0; desmat[25][4]= 0;
        desmat[26][2]= 0; desmat[26][3]= 0; desmat[26][4]= 0;
        desmat[27][2]= 0; desmat[27][3]= 0; desmat[27][4]= 0;

        desmat[1][5] = 0;
        desmat[2][5] = 0;
        desmat[3][5] = 0;
        desmat[4][5] = 0;
        desmat[5][5] =-1;
        desmat[6][5] =-1;
        desmat[7][5] = 1;
        desmat[8][5] = 1;
        desmat[9][5] =-1;
        desmat[10][5]=-1;
        desmat[11][5]= 1;
        desmat[12][5]= 1;
        desmat[13][5]= 0;
        desmat[14][5]= 0;
        desmat[15][5]= 0;
        desmat[16][5]= 0;
        desmat[17][5]= 0;
        desmat[18][5]= 0;
        desmat[19][5]= 0;
        desmat[20][5]= 0;
        desmat[21][5]=-1;
        desmat[22][5]=-1;
        desmat[23][5]= 1;
        desmat[24][5]= 1;
        desmat[25][5]= 0;
        desmat[26][5]= 0;
        desmat[27][5]= 0;
       }
}
```

```
/* CREATE BOX-BEHNKEN CORE DESIGN */
int box5(facs,totrow)
{
int i, j, col;

    for (i=1; i<=totrow; i++)
        desmat[i][1] = 1;

   if (facs == 5)
      { desmat[1][2] =-1; desmat[1][3] =-1; desmat[1][4] = 0;
        desmat[2][2] = 1; desmat[2][3] =-1; desmat[2][4] = 0;
        desmat[3][2] =-1; desmat[3][3] = 1; desmat[3][4] = 0;
        desmat[4][2] = 1; desmat[4][3] = 1; desmat[4][4] = 0;
        desmat[5][2] = 0; desmat[5][3] = 0; desmat[5][4] =-1;
        desmat[6][2] = 0; desmat[6][3] = 0; desmat[6][4] = 1;
        desmat[7][2] = 0; desmat[7][3] = 0; desmat[7][4] =-1;
        desmat[8][2] = 0; desmat[8][3] = 0; desmat[8][4] = 1;
        desmat[9][2] = 0; desmat[9][3] =-1; desmat[9][4] = 0;
        desmat[10][2]= 0; desmat[10][3]= 1; desmat[10][4]= 0;
        desmat[11][2]= 0; desmat[11][3]=-1; desmat[11][4]= 0;
        desmat[12][2]= 0; desmat[12][3]= 1; desmat[12][4]= 0;
        desmat[13][2]=-1; desmat[13][3]= 0; desmat[13][4]=-1;
        desmat[14][2]= 1; desmat[14][3]= 0; desmat[14][4]=-1;
        desmat[15][2]=-1; desmat[15][3]= 0; desmat[15][4]= 1;
        desmat[16][2]= 1; desmat[16][3]= 0; desmat[16][4]= 1;
        desmat[17][2]= 0; desmat[17][3]= 0; desmat[17][4]= 0;
        desmat[18][2]= 0; desmat[18][3]= 0; desmat[18][4]= 0;
        desmat[19][2]= 0; desmat[19][3]= 0; desmat[19][4]= 0;
        desmat[20][2]= 0; desmat[20][3]= 0; desmat[20][4]= 0;
        desmat[21][2]= 0; desmat[21][3]=-1; desmat[21][4]=-1;
        desmat[22][2]= 0; desmat[22][3]= 1; desmat[22][4]=-1;
        desmat[23][2]= 0; desmat[23][3]=-1; desmat[23][4]= 1;
        desmat[24][2]= 0; desmat[24][3]= 1; desmat[24][4]= 1;
        desmat[25][2]=-1; desmat[25][3]= 0; desmat[25][4]= 0;
        desmat[26][2]= 1; desmat[26][3]= 0; desmat[26][4]= 0;
        desmat[27][2]=-1; desmat[27][3]= 0; desmat[27][4]= 0;
        desmat[28][2]= 1; desmat[28][3]= 0; desmat[28][4]= 0;
        desmat[29][2]= 0; desmat[29][3]= 0; desmat[29][4]=-1;
        desmat[30][2]= 0; desmat[30][3]= 0; desmat[30][4]= 1;
        desmat[31][2]= 0; desmat[31][3]= 0; desmat[31][4]=-1;
        desmat[32][2]= 0; desmat[32][3]= 0; desmat[32][4]= 1;
        desmat[33][2]=-1; desmat[33][3]= 0; desmat[33][4]= 0;
        desmat[34][2]= 1; desmat[34][3]= 0; desmat[34][4]= 0;
        desmat[35][2]=-1; desmat[35][3]= 0; desmat[35][4]= 0;
        desmat[36][2]= 1; desmat[36][3]= 0; desmat[36][4]= 0;
        desmat[37][2]= 0; desmat[37][3]=-1; desmat[37][4]= 0;
        desmat[38][2]= 0; desmat[38][3]= 1; desmat[38][4]= 0;
        desmat[39][2]= 0; desmat[39][3]=-1; desmat[39][4]= 0;
        desmat[40][2]= 0; desmat[40][3]= 1; desmat[40][4]= 0;
        desmat[41][2]= 0; desmat[41][3]= 0; desmat[41][4]= 0;
```

```
desmat[42][2]= 0; desmat[42][3]= 0; desmat[42][4]= 0;
desmat[43][2]= 0; desmat[43][3]= 0; desmat[43][4]= 0;
desmat[44][2]= 0; desmat[44][3]= 0; desmat[44][4]= 0;
desmat[45][2]= 0; desmat[45][3]= 0; desmat[45][4]= 0;
desmat[46][2]= 0; desmat[46][3]= 0; desmat[46][4]= 0;

desmat[1][5] = 0; desmat[1][6] = 0;
desmat[2][5] = 0; desmat[2][6] = 0;
desmat[3][5] = 0; desmat[3][6] = 0;
desmat[4][5] = 0; desmat[4][6] = 0;
desmat[5][5] =-1; desmat[5][6] = 0;
desmat[6][5] =-1; desmat[6][6] = 0;
desmat[7][5] = 1; desmat[7][6] = 0;
desmat[8][5] = 1; desmat[8][6] = 0;
desmat[9][5] = 0; desmat[9][6] =-1;
desmat[10][5]= 0; desmat[10][6]=-1;
desmat[11][5]= 0; desmat[11][6]= 1;
desmat[12][5]= 0; desmat[12][6]= 1;
desmat[13][5]= 0; desmat[13][6]= 0;
desmat[14][5]= 0; desmat[14][6]= 0;
desmat[15][5]= 0; desmat[15][6]= 0;
desmat[16][5]= 0; desmat[16][6]= 0;
desmat[17][5]=-1; desmat[17][6]=-1;
desmat[18][5]= 1; desmat[18][6]=-1;
desmat[19][5]=-1; desmat[19][6]= 1;
desmat[20][5]= 1; desmat[20][6]= 1;
desmat[21][5]= 0; desmat[21][6]= 0;
desmat[22][5]= 0; desmat[22][6]= 0;
desmat[23][5]= 0; desmat[23][6]= 0;
desmat[24][5]= 0; desmat[24][6]= 0;
desmat[25][5]=-1; desmat[25][6]= 0;
desmat[26][5]=-1; desmat[26][6]= 0;
desmat[27][5]= 1; desmat[27][6]= 0;
desmat[28][5]= 1; desmat[28][6]= 0;
desmat[29][5]= 0; desmat[29][6]=-1;
desmat[30][5]= 0; desmat[30][6]=-1;
desmat[31][5]= 0; desmat[31][6]= 1;
desmat[32][5]= 0; desmat[32][6]= 1;
desmat[33][5]= 0; desmat[33][6]=-1;
desmat[34][5]= 0; desmat[34][6]=-1;
desmat[35][5]= 0; desmat[35][6]= 1;
desmat[36][5]= 0; desmat[36][6]= 1;
desmat[37][5]=-1; desmat[37][6]= 0;
desmat[38][5]=-1; desmat[38][6]= 0;
desmat[39][5]= 1; desmat[39][6]= 0;
desmat[40][5]= 1; desmat[40][6]= 0;
desmat[41][5]= 0; desmat[41][6]= 0;
desmat[42][5]= 0; desmat[42][6]= 0;
desmat[43][5]= 0; desmat[43][6]= 0;
desmat[44][5]= 0; desmat[44][6]= 0;
```

```
        desmat[45][5]= 0; desmat[45][6]= 0;
        desmat[46][5]= 0; desmat[46][6]= 0;
       }
}

/* CREATE BOX-BEHNKEN CORE DESIGN */
int box6(facs,totrow)
{
int i, j, col;

    for (i=1; i<=totrow; i++)
        desmat[i][1] = 1;

     if (facs == 6)
      { desmat[1][2] =-1; desmat[1][3] =-1; desmat[1][4] = 0;
        desmat[2][2] = 1; desmat[2][3] =-1; desmat[2][4] = 0;
        desmat[3][2] =-1; desmat[3][3] = 1; desmat[3][4] = 0;
        desmat[4][2] = 1; desmat[4][3] = 1; desmat[4][4] = 0;
        desmat[5][2] =-1; desmat[5][3] =-1; desmat[5][4] = 0;
        desmat[6][2] = 1; desmat[6][3] =-1; desmat[6][4] = 0;
        desmat[7][2] =-1; desmat[7][3] = 1; desmat[7][4] = 0;
        desmat[8][2] = 1; desmat[8][3] = 1; desmat[8][4] = 0;
        desmat[9][2] = 0; desmat[9][3] =-1; desmat[9][4] =-1;
        desmat[10][2]= 0; desmat[10][3]= 1; desmat[10][4]=-1;
        desmat[11][2]= 0; desmat[11][3]=-1; desmat[11][4]= 1;
        desmat[12][2]= 0; desmat[12][3]= 1; desmat[12][4]= 1;
        desmat[13][2]= 0; desmat[13][3]=-1; desmat[13][4]=-1;
        desmat[14][2]= 0; desmat[14][3]= 1; desmat[14][4]=-1;
        desmat[15][2]= 0; desmat[15][3]=-1; desmat[15][4]= 1;
        desmat[16][2]= 0; desmat[16][3]= 1; desmat[16][4]= 1;
        desmat[17][2]= 0; desmat[17][3]= 0; desmat[17][4]=-1;
        desmat[18][2]= 0; desmat[18][3]= 0; desmat[18][4]= 1;
        desmat[19][2]= 0; desmat[19][3]= 0; desmat[19][4]=-1;
        desmat[20][2]= 0; desmat[20][3]= 0; desmat[20][4]= 1;
        desmat[21][2]= 0; desmat[21][3]= 0; desmat[21][4]=-1;
        desmat[22][2]= 0; desmat[22][3]= 0; desmat[22][4]= 1;
        desmat[23][2]= 0; desmat[23][3]= 0; desmat[23][4]=-1;
        desmat[24][2]= 0; desmat[24][3]= 0; desmat[24][4]= 1;
        desmat[25][2]=-1; desmat[25][3]= 0; desmat[25][4]= 0;
        desmat[26][2]= 1; desmat[26][3]= 0; desmat[26][4]= 0;
        desmat[27][2]=-1; desmat[27][3]= 0; desmat[27][4]= 0;
        desmat[28][2]= 1; desmat[28][3]= 0; desmat[28][4]= 0;
        desmat[29][2]=-1; desmat[29][3]= 0; desmat[29][4]= 0;
        desmat[30][2]= 1; desmat[30][3]= 0; desmat[30][4]= 0;
        desmat[31][2]=-1; desmat[31][3]= 0; desmat[31][4]= 0;
        desmat[32][2]= 1; desmat[32][3]= 0; desmat[32][4]= 0;
        desmat[33][2]= 0; desmat[33][3]=-1; desmat[33][4]= 0;
        desmat[34][2]= 0; desmat[34][3]= 1; desmat[34][4]= 0;
        desmat[35][2]= 0; desmat[35][3]=-1; desmat[35][4]= 0;
        desmat[36][2]= 0; desmat[36][3]= 1; desmat[36][4]= 0;
```

```
desmat[37][2]= 0; desmat[37][3]=-1; desmat[37][4]= 0;
desmat[38][2]= 0; desmat[38][3]= 1; desmat[38][4]= 0;
desmat[39][2]= 0; desmat[39][3]=-1; desmat[39][4]= 0;
desmat[40][2]= 0; desmat[40][3]= 1; desmat[40][4]= 0;
desmat[41][2]=-1; desmat[41][3]= 0; desmat[41][4]=-1;
desmat[42][2]= 1; desmat[42][3]= 0; desmat[42][4]=-1;
desmat[43][2]=-1; desmat[43][3]= 0; desmat[43][4]= 1;
desmat[44][2]= 1; desmat[44][3]= 0; desmat[44][4]= 1;
desmat[45][2]=-1; desmat[45][3]= 0; desmat[45][4]=-1;
desmat[46][2]= 1; desmat[46][3]= 0; desmat[46][4]=-1;
desmat[47][2]=-1; desmat[47][3]= 0; desmat[47][4]= 1;
desmat[48][2]= 1; desmat[48][3]= 0; desmat[48][4]= 1;
desmat[49][2]= 0; desmat[49][3]= 0; desmat[49][4]= 0;
desmat[50][2]= 0; desmat[50][3]= 0; desmat[50][4]= 0;
desmat[51][2]= 0; desmat[51][3]= 0; desmat[51][4]= 0;
desmat[52][2]= 0; desmat[52][3]= 0; desmat[52][4]= 0;
desmat[53][2]= 0; desmat[53][3]= 0; desmat[53][4]= 0;
desmat[54][2]= 0; desmat[54][3]= 0; desmat[54][4]= 0;

desmat[1][5] =-1; desmat[1][6] = 0; desmat[1][7] = 0;
desmat[2][5] =-1; desmat[2][6] = 0; desmat[2][7] = 0;
desmat[3][5] =-1; desmat[3][6] = 0; desmat[3][7] = 0;
desmat[4][5] =-1; desmat[4][6] = 0; desmat[4][7] = 0;
desmat[5][5] = 1; desmat[5][6] = 0; desmat[5][7] = 0;
desmat[6][5] = 1; desmat[6][6] = 0; desmat[6][7] = 0;
desmat[7][5] = 1; desmat[7][6] = 0; desmat[7][7] = 0;
desmat[8][5] = 1; desmat[8][6] = 0; desmat[8][7] = 0;
desmat[9][5] = 0; desmat[9][6] =-1; desmat[9][7] = 0;
desmat[10][5]= 0; desmat[10][6]=-1; desmat[10][7]= 0;
desmat[11][5]= 0; desmat[11][6]=-1; desmat[11][7]= 0;
desmat[12][5]= 0; desmat[12][6]=-1; desmat[12][7]= 0;
desmat[13][5]= 0; desmat[13][6]= 1; desmat[13][7]= 0;
desmat[14][5]= 0; desmat[14][6]= 1; desmat[14][7]= 0;
desmat[15][5]= 0; desmat[15][6]= 1; desmat[15][7]= 0;
desmat[16][5]= 0; desmat[16][6]= 1; desmat[16][7]= 0;
desmat[17][5]=-1; desmat[17][6]= 0; desmat[17][7]=-1;
desmat[18][5]=-1; desmat[18][6]= 0; desmat[18][7]=-1;
desmat[19][5]= 1; desmat[19][6]= 0; desmat[19][7]=-1;
desmat[20][5]= 1; desmat[20][6]= 0; desmat[20][7]=-1;
desmat[21][5]=-1; desmat[21][6]= 0; desmat[21][7]= 1;
desmat[22][5]=-1; desmat[22][6]= 0; desmat[22][7]= 1;
desmat[23][5]= 1; desmat[23][6]= 0; desmat[23][7]= 1;
desmat[24][5]= 1; desmat[24][6]= 0; desmat[24][7]= 1;
desmat[25][5]=-1; desmat[25][6]=-1; desmat[25][7]= 0;
desmat[26][5]=-1; desmat[26][6]=-1; desmat[26][7]= 0;
desmat[27][5]= 1; desmat[27][6]=-1; desmat[27][7]= 0;
desmat[28][5]= 1; desmat[28][6]=-1; desmat[28][7]= 0;
desmat[29][5]=-1; desmat[29][6]= 1; desmat[29][7]= 0;
desmat[30][5]=-1; desmat[30][6]= 1; desmat[30][7]= 0;
desmat[31][5]= 1; desmat[31][6]= 1; desmat[31][7]= 0;
```

```
        desmat[32][5]= 1; desmat[32][6]= 1; desmat[32][7]= 0;
        desmat[33][5]= 0; desmat[33][6]=-1; desmat[33][7]=-1;
        desmat[34][5]= 0; desmat[34][6]=-1; desmat[34][7]=-1;
        desmat[35][5]= 0; desmat[35][6]= 1; desmat[35][7]=-1;
        desmat[36][5]= 0; desmat[36][6]= 1; desmat[36][7]=-1;
        desmat[37][5]= 0; desmat[37][6]=-1; desmat[37][7]= 1;
        desmat[38][5]= 0; desmat[38][6]=-1; desmat[38][7]= 1;
        desmat[39][5]= 0; desmat[39][6]= 1; desmat[39][7]= 1;
        desmat[40][5]= 0; desmat[40][6]= 1; desmat[40][7]= 1;
        desmat[41][5]= 0; desmat[41][6]= 0; desmat[41][7]=-1;
        desmat[42][5]= 0; desmat[42][6]= 0; desmat[42][7]=-1;
        desmat[43][5]= 0; desmat[43][6]= 0; desmat[43][7]=-1;
        desmat[44][5]= 0; desmat[44][6]= 0; desmat[44][7]=-1;
        desmat[45][5]= 0; desmat[45][6]= 0; desmat[45][7]= 1;
        desmat[46][5]= 0; desmat[46][6]= 0; desmat[46][7]= 1;
        desmat[47][5]= 0; desmat[47][6]= 0; desmat[47][7]= 1;
        desmat[48][5]= 0; desmat[48][6]= 0; desmat[48][7]= 1;
        desmat[49][5]= 0; desmat[49][6]= 0; desmat[49][7]= 0;
        desmat[50][5]= 0; desmat[50][6]= 0; desmat[50][7]= 0;
        desmat[51][5]= 0; desmat[51][6]= 0; desmat[51][7]= 0;
        desmat[52][5]= 0; desmat[52][6]= 0; desmat[52][7]= 0;
        desmat[53][5]= 0; desmat[53][6]= 0; desmat[53][7]= 0;
        desmat[54][5]= 0; desmat[54][6]= 0; desmat[54][7]= 0;
      }
}

/* FUNCTION LABEL ASSIGNS LABELS TO MATRIX COLUMNS */

int label(facs)

{  int i,j,k,l,m,p,col,col2;

col = facs + 1;
col2 = 2*facs + 1;

desmat[0][1] = 0;

for (j=2; j<=col; j++)
   { desmat[0][j] = j-1;
     desmat[0][j+facs] = 11*(j-1);
   }


   if (facs>=2)   desmat[0][col2+1] = 12;

   if (facs>=3)

         {desmat[0][col2+2] = 13;
          desmat[0][col2+3] = 23;
          desmat[0][col2+4] = 123;
```

```
      }

if (facs>=4)

      {desmat[0][col2+5] = 14;
       desmat[0][col2+6] = 24;
       desmat[0][col2+7] = 124;
       desmat[0][col2+8] = 34;
       desmat[0][col2+9] = 134;
       desmat[0][col2+10]= 234;
       desmat[0][col2+11]= 1234;
      }

if (facs>=5)

      {desmat[0][col2+12] = 15;
       desmat[0][col2+13] = 25;
       desmat[0][col2+14] = 125;
       desmat[0][col2+15] = 35;
       desmat[0][col2+16] = 135;
       desmat[0][col2+17] = 235;
       desmat[0][col2+18] = 1235;
       desmat[0][col2+19] = 45;
       desmat[0][col2+20] = 145;
       desmat[0][col2+21] = 245;
       desmat[0][col2+22] = 1245;
       desmat[0][col2+23] = 345;
       desmat[0][col2+24] = 1345;
       desmat[0][col2+25] = 2345;
       desmat[0][col2+26] = 12345;
      }

if (facs>=6)

      {desmat[0][col2+27] = 16;
       desmat[0][col2+28] = 26;
       desmat[0][col2+29] = 126;
       desmat[0][col2+30] = 36;
       desmat[0][col2+31] = 136;
       desmat[0][col2+32]=  236;
       desmat[0][col2+33]=  1236;
       desmat[0][col2+34] = 46;
       desmat[0][col2+35] = 146;
       desmat[0][col2+36] = 246;
       desmat[0][col2+37] = 1246;
       desmat[0][col2+38] = 346;
       desmat[0][col2+39] = 1346;
       desmat[0][col2+40] = 2346;
       desmat[0][col2+41] = 12346;
       desmat[0][col2+42] = 56;
```

```
            desmat[0][col2+43] = 156;
            desmat[0][col2+44] = 256;
            desmat[0][col2+45] = 1256;
            desmat[0][col2+46] = 356;
            desmat[0][col2+47] = 1356;
            desmat[0][col2+48] = 2356;
            desmat[0][col2+49] = 12356;
            desmat[0][col2+50] = 456;
            desmat[0][col2+51] = 1456;
            desmat[0][col2+52] = 2456;
            desmat[0][col2+53] = 12456;
            desmat[0][col2+54] = 3456;
            desmat[0][col2+55] = 13456;
            desmat[0][col2+56] = 23456;
            desmat[0][col2+57] = 123456;
           }
}

/* FUNCTION boxLABEL ASSIGNS LABELS TO BOX-BEHNKEN MATRIX COLUMNS
*/

int boxlabel(facs)

{  int i,j,k,l,m,p,col,col2;

col = facs + 1;
col2 = 2*facs + 1;

desmat[0][1] = 0;

for (j=2; j<=col; j++)
   { desmat[0][j] = j-1;
     desmat[0][j+facs] = 11*(j-1);
   }


   if (facs>=2)   desmat[0][col2+1] = 12;

   if (facs>=3)

           {desmat[0][col2+2] = 13;
            desmat[0][col2+3] = 23;
           }

   if (facs>=4)

           {desmat[0][col2+4] = 14;
            desmat[0][col2+5] = 24;
            desmat[0][col2+6] = 34;
           }
```

```
    if (facs>=5)

            {desmat[0][col2+7] =  15;
             desmat[0][col2+8] =  25;
             desmat[0][col2+9] =  35;
             desmat[0][col2+10] = 45;
            }

    if (facs>=6)

            {desmat[0][col2+11] = 16;
             desmat[0][col2+12] = 26;
             desmat[0][col2+13] = 36;
             desmat[0][col2+14] = 46;
             desmat[0][col2+15] = 56;
            }
}
```

FACSET.PRG

```
*
* FACSET.PRG
*
PROCEDURE p_facset
PARAMETERS level,num_rows
num_fields = 3
retrieve = .F.
data_entry = .F.
factor_selection = .F.
exit_facset = .F.
save_facset_screen = savescreen(2,9,8,26)
DO WHILE !exit_facset
  prompt_size = 16
  @ 2,9,8,26 BOX sl_box
  @ 24,0 SAY SPACE(80)
  @ 3,10 PROMPT menu_pad("Save",prompt_size);
         MESSAGE "SELECT TO SAVE THE CURRENT DATA ENTRY
SESSION"
  @ 4,10 PROMPT menu_pad("Retrieve",prompt_size);
         MESSAGE "SELECT TO RETRIEVE A PREVIOUSLY SAVED
FILE"
  @ 5,10 PROMPT menu_pad("Data Entry",prompt_size);
         MESSAGE "SELECT TO ENTER NEW DATA OR EDIT RETRIEVED
DATA"
  @ 6,10 PROMPT menu_pad("Factor Selection",prompt_size);
         MESSAGE "SELECT TO RETAIN FACTORS FOR FUTHER
ANALYSIS"
  @ 7,10 PROMPT menu_pad("Exit",prompt_size);
         MESSAGE "SELECT TO EXIT THIS MENU"
  choice = 1
  MENU TO choice
  DO CASE
     CASE choice = 1
      IF (retrieve.OR.data_entry)
        * get the file name of the save file
        overwrite = .F.
        prompt_msg = "INPUT THE FACTOR SETTINGS FILE NAME"
        filename =
input_filename(".FAC",@overwrite,prompt_msg)
        IF overwrite
           DO p_write_it  && write the data to disk using
the filename
        ENDIF
      ELSE
        @ 0,1 SAY "Error -- You must have DATA present" + ;
                   " to SAVE a file <PRESS RETURN>"
        key_wait = inkey(0)
        @ 0,0 SAY SPACE(80)
```

```
      ENDIF
     CASE choice = 2
      retrieve = .F.  && for use in the Factor Selection
menu item
      @ 0,1 SAY "INPUT THE FACTOR SETTINGS FILE NAME"
      filename = get_filename("*.FAC")
      @ 0,0 SAY SPACE(80)
      IF FILE(filename)
        DO p_read_it            && read the saved data from
file
        retrieve = .T.
      ELSE
        @ 0,1 SAY "Error -- This file does not exist" + ;
                   " <PRESS RETURN>"
        key_wait = inkey(0)
        @ 0,0 SAY SPACE(80)
      ENDIF
    CASE choice = 3
      IF p_data_entry() = 0          && operate on the data

        data_entry = .F.             && abnormal ending
      ELSE
        data_entry = .T.
      ENDIF
    CASE choice = 4               && select the correct
factors
      *  a file must exist to select factors
      IF retrieve
        factor_selection = .F.
        DECLARE ar[num_rows]
        FOR i = 1 TO num_rows
          ar[i] = " "
          ar[i] = "  " + chr(186) + name[i] + " " +
ltrim(str(low[i])) + ;
                  " " + ltrim(str(high[i])) + " " +
vlabel[i]
        NEXT
        savebrow = savescreen(11,10,18,62)
        afill(asterisk,.F.)  && fill with false values (no
selection)
        sel = abrowse(ar,num_rows,END_KEY,11,10,18,62)
        restscreen(11,10,18,62,savebrow)
        IF sel > 0  && ESCAPE KEY NOT PRESSED IN SELECTION
          factor_selection = .T.
        ENDIF
      ELSE
        @ 0,1 SAY "Error -- You must have a file retrieved"
+ ;
                   " to SELECT from <PRESS RETURN>"
        key_wait = inkey(0)
```

```
      @ 0,0 SAY SPACE(80)
    ENDIF
  CASE choice = 5
    exit_facset = .T.
  ENDCASE
ENDDO
*  restore the screen for the facset screen
restscreen(2,9,8,26,save_facset_screen)
RETURN

PROCEDURE p_write_it
  *
  * DETERMINE THE NUMBER OF FACTORS REMAINING FROM FACTOR
SELECTION
  *
  num_factors = 0
  IF factor_selection
    FOR i = 1 to num_rows
      IF asterisk[i]
        num_factors = num_factors + 1
      ENDIF
    NEXT
  ELSE
    num_factors = num_rows
  ENDIF
  * DO NOT ALLOW A FILE TO BE WRITTEN IF THERE ARE NO
FACTORS SELECTED
  IF factor_selection.AND.num_factors = 0
    @ 0,1 SAY "NO FILE WRITTEN BECAUSE NO FACTORS WERE
SELECTED <RETURN>"
    wait_key = inkey(0)
    @ 0,0 SAY SPACE(80)
    factor_selection = .F.
    RETURN
  ENDIF

  handle = FCREATE(filename)
  eol = chr(13) + chr(10)
  comma = ","
  wbuffer = ltrim(str(num_factors)) + eol
  FWRITE(handle,wbuffer)
  wbuffer = ""
  FOR i = 1 TO num_rows
    IF factor_selection
      write_value = asterisk[i]
    ELSE
      write_value = .T.
    ENDIF
    IF write_value     && IF TRUE THEN ALLOW THE WRITE TO
TAKE PLACE
```

```
      wbuffer = ltrim(str(low[i])) + comma + ;
                ltrim(str(high[i])) + comma + vlabel[i]
      FWRITE(handle,wbuffer)
      IF i < num_rows
        FWRITE(handle,eol)
      ENDIF
    ENDIF
    wbuffer = ""
  NEXT
  factor_selection = .F.
FCLOSE(handle)
RETURN

PROCEDURE p_read_it
PRIVATE parse_it[3]
LINE_SIZE = 512
line = SPACE(LINE_SIZE)
comma = ","
afill(parse_it," ")
handle = FOPEN(filename)
first_read = .T.
num_fields = 1
DO WHILE freadln(handle,@line,LINE_SIZE)
 buffer = line
  FOR i = 1 to num_fields
    comma_pos = AT(comma,buffer)
    parse_it[i] = substr(buffer,1,(comma_pos - 1))
    buffer = substr(buffer,-(len(buffer) - comma_pos))
  NEXT
  IF first_read
    num_rows = val(buffer)
    PUBLIC
name[num_rows],low[num_rows],med[num_rows],high[num_rows]
    PUBLIC vlabel[num_rows],asterisk[num_rows]
    num_fields = 2
    j = 0
    first_read = .F.
  ELSE
    j = j + 1
    name[j]   = "X" + ltrim(str(j))
    low[j]    = val(parse_it[1])
    high[j]   = val(parse_it[2])
    vlabel[j] = trim(buffer)
  ENDIF
ENDDO
FCLOSE(handle)
RETURN

FUNCTION p_data_entry
@ 0,1 SAY "SELECT OR ENTER A DESIGN FILENAME"
```

```
des_filename = get_filename("*.DES")
@ 0,0 SAY SPACE(80)
IF !FILE(des_filename)
  @ 0,1 SAY " ERROR -- DESIGN FILE DOES NOT EXIST " + ;
             "PRESS ANY KEY TO CONTINUE"
  wait_key = inkey(0)
  @ 0,0 SAY SPACE(80)
  RETURN(0)
ELSE
  DO p_flagval WITH des_filename
  group_num = flag_val[1]
  num_rows  = flag_val[2]    && get the number of factors
ENDIF
@ 0,1 SAY "RETURN TO ENTER DATA - ARROWS TO MOVE - END TO
LEAVE"
PUBLIC
name[num_rows],low[num_rows],med[num_rows],high[num_rows]
PUBLIC vlabel[num_rows],asterisk[num_rows]
afill(name,SPACE(20))
afill(low,0)
afill(med,0)
afill(high,0)
afill(vlabel,SPACE(20))
afill(asterisk,.F.)  && fill with false values (no
selection)
pos = 1
FOR i = 1 TO num_rows
  name[i] = "X" + ltrim(str(i))
NEXT
current_field = 1
current_row = 1
key = 0
* save the area around the input area
savewindow = savescreen(18,1,22,63)
@ 18,1 SAY CHR(213) + REPLICATE(CHR(205),61) + CHR(184)
@ 19,1 SAY CHR(179) + " VARIABLE NAME           LOW
HIGH" + ;
           "      VARIABLE LABEL   " + CHR(179)
@ 20,1 TO 22,63  && Draw a box around input field
@ 20,1 SAY CHR(195)  && LEFT CORNER BOX EXTENDER
@ 20,63 SAY CHR(180) && RIGHT CORNER BOX EXTENDER
*@ 10,50 SAY SPACE(10)  && Fill in the block with spaces
DO WHILE key != END_KEY
  key = 0
  SET CURSOR OFF
  * select cell to edit or escape
  DO WHILE (key != ENTER.AND.key != END_KEY)
    @ 21,2 SAY menu_pad(name[current_row],21)
    FOR field_num = 1 TO num_fields
      SET COLOR TO &norm
```

```
      @ 20,23+(field_num-1) * 10 SAY chr(194)
      @ 21,23+(field_num-1) * 10 SAY chr(179)
      @ 22,23+(field_num-1) * 10 SAY chr(193)
      DO get_string_proc
      IF field_num = current_field
        SET COLOR TO &enh
      ELSE
        SET COLOR TO &norm
      ENDIF
     @ 21,24+(field_num-1) * 10 SAY &array_name[current_row]
;
        PICT pic_string
    NEXT

    key = inkey(0)   && Wait until a key is pressed
    * ESC PRESSED THEN EXIT AND RETURN 0
    IF key = ESC
      @ 0,0 SAY SPACE(80)
      restscreen(18,1,22,63,savewindow)
      RETURN(0)
    ENDIF
    * Remove the highlight from the current field
    SET COLOR TO &norm
    @ 21,24+(current_field-1) * 10 SAY
&array_name[current_row] ;
      PICT pic_string

    DO CASE
      CASE key = DOWN_ARROW
        DO dn_arrow_proc
      CASE key = UP_ARROW
        DO up_arrow_proc
      CASE key = LT_ARROW
        DO lt_arrow_proc
      CASE key = RT_ARROW
        DO rt_arrow_proc
      CASE ((key >= asc('0').AND.key <= asc('9')).OR. ;
            (key >= asc('A').AND.key <= asc('z')))
        EXIT
    ENDCASE
  ENDDO
  SET CURSOR ON
  IF key <> END_KEY
    * starting edit mode
    IF key != ENTER
      KEYBOARD chr(key)
    ENDIF
    field_num = current_field
    DO get_string_proc
    get_string = &array_name[current_row]
```

```
    @ 21,24+(current_field-1)*10 GET get_string PICT
pic_string
    READ
    IF UPDATED()  && if there was a change in the string
then update
      &array_name[current_row] = get_string
    ENDIF
  ENDIF
ENDDO
SET CURSOR ON
@ 0,0 SAY SPACE(80)
restscreen(18,1,22,63,savewindow)
RETURN(1)  && IF NORMAL END RETURN 1

PROCEDURE get_string_proc
  PUBLIC array_name,pic_string
  pic_string = "@S10@B 999999999999999.9999"
  DO CASE
    CASE field_num = 1
      array_name = "low"
    CASE field_num = 2
        array_name = "high"
    CASE field_num = 3
        array_name = "vlabel"
        pic_string = "XXXXXXXXXXXXXXXXXXX"
  ENDCASE
RETURN


PROCEDURE dn_arrow_proc
  * If not on last line go down by fields_line
  IF current_row < num_rows
    current_row = current_row + 1
  ENDIF
RETURN

PROCEDURE up_arrow_proc
  * If not on top line
  IF current_row > 1
    current_row = current_row - 1
  ENDIF
RETURN

PROCEDURE lt_arrow_proc
  * If not at first field get...
  IF current_field > 1
    current_field = current_field - 1
  ELSEIF current_row > 1    && current_field = 1
    * Go to last column on previous row
    current_field = num_fields
```

```
    current_row = current_row - 1
  ENDIF
RETURN

PROCEDURE rt_arrow_proc
  * If not at last field get...
  IF current_field < num_fields
    current_field = current_field + 1
  ELSEIF current_row < num_rows    && current_field = 1
    * Go to first column on next row
    current_field = 1
    current_row = current_row + 1
  ENDIF
RETURN
```

RAWDAT.PRG

```
PROCEDURE p_rawdat
       @ 0,1 SAY "SELECT OR ENTER A DESIGN FILENAME"
          des_filename = get_filename("*.DES")
       @ 0,0 SAY SPACE(80)
          IF !FILE(des_filename)
         @ 0,1 SAY " ERROR -- DESIGN FILE DOES NOT EXIST
<RETURN>"
            wait_key = inkey(0)
         @ 0,0 SAY SPACE(80)
         RETURN
       ENDIF

       * make sure that exp.des is current filename
       IF UPPER(des_filename) != "EXP.DES"
         COPY FILE &des_filename TO exp.des
       ENDIF

       * READ .DES FOR # OF GROUPS AND # OF FACTORS
       DO p_flagval WITH des_filename
       groups      = flag_val[1]
       num_factors = flag_val[2]
       IF FILE("exp.fac")
         ERASE
       ENDIF
       @ 0,1 SAY "SELECT OR ENTER A FACTORS FILENAME"
       fac_filename = get_filename("*.FAC")
       @ 0,0 SAY SPACE(80)
       IF !FILE(fac_filename)
         @ 0,1 SAY " ERROR -- FACTORS FILE DOES NOT EXIST
<RETURN>"
         wait_key = inkey(0)
         @ 0,0 SAY SPACE(80)
         RETURN
```

```
        ENDIF

        * check to see that .des and .fac have the same
number of factors
        handle = FOPEN(fac_filename)
        fnum_factors = val(freadline(handle))
        FCLOSE(handle)
        IF fnum_factors != num_factors
          @ 0,1 SAY " ERROR -- NOT THE SAME NUMBER OF FACTORS
IN .DES " + ;
                    " AND .FAC SELECTED <RETURN>"
          wait_key = inkey(0)
          @ 0,0 SAY SPACE(80)
          RETURN
        ENDIF

        COPY FILE &fac_filename TO exp.fac
        IF groups > 0  && GROUP SCREENING
          DECLARE group_num[num_factors]  && create group
number array
          afill(group_num,0)
          @ 24,0 SAY SPACE(80)
          FOR i = 1 TO num_factors
            @ 24,0 SAY "Enter the group number for factor " +
;
                "X" + ltrim(str(i)) + " ==> " ;
            GET group_num[i] PICT "999" ;
            VALID ((group_num[i] > 0).AND.(group_num[i] <=
groups))
            READ
            IF LASTKEY() = ESC
           @ 0,0 SAY SPACE(80)
           @ 24,0 SAY SPACE(80)
           RETURN
            ENDIF
          NEXT
          * write out the group number file
          handle = FCREATE("exp.grp")
          FOR i = 1 TO num_factors
            buffer = LTRIM(STR(group_num[i])) + " "
            FWRITE(handle,@buffer)
          NEXT
          FCLOSE(handle)
          @ 24,0 SAY SPACE(80)
        ENDIF  && for group screening

        IF FILE("clipraw.exe")
          * C CODE TO GROUP RAW DATA FILE
          SAVE SCREEN
          CLEAR
```

```
          RUN clipraw
          RESTORE SCREEN
          overwrite = .F.
          prompt_msg = "PLEASE ENTER THE RAW DATA FILE NAME"
          raw_filename =
input_filename(".RAW",@overwrite,prompt_msg)
          IF !EMPTY(raw_filename) .AND. overwrite
            COPY FILE exp.raw TO &raw_filename
          ENDIF
        ELSE
          @ 0,1 SAY " ERROR -- FILE > clipraw.exe < DOES NOT
EXIST <RETURN>"
          wait_key = inkey(0)
          @ 0,0 SAY SPACE(80)
          RETURN
        ENDIF
RETURN
```

CLIPRAW.EXE

```
/*  PROGRAM CLIPRAW.C -- A FUNCTION FOR CLIPPER */

# include <stdlib.h>
# include <stdio.h>
# include <math.h>
# define LIM 35

main()
{
FILE *infile,*outfile;
int return_value;
int i,j,k,l;      /* COUNTERS */
int m,n;          /* # ROWS AND # COLUMNS */
int c;            /* # DIFFERENT LEVELS */
int orthog;       /* ORTHOGONAL DESIGN FLAG <1> IS ORTHOGONAL
*/
int choice;       /* CHOOSE THE DESIRED TRANSFORMATION */
static float low[LIM],high[LIM];
static float mid[LIM],semirange[LIM];
static float x[LIM][LIM],var_label[LIM];
int level,row,facs,reps,creps,gpscreen,factors,fac_factors;
char ch;

/*  GROUP DECLARATIONS  */
static int factor_groupnum[LIM];
static int group_table[LIM][LIM];
static int num_factors_ingroup[LIM];
static int factor_num;
```

```
/* READ FROM FILE EXP.DES */
infile = fopen("exp.des","r");

fscanf(infile,"%d",&gpscreen);
fscanf(infile,"%d",&factors);
printf("GROUP %d FACTOR %d\n",gpscreen,factors);
fscanf(infile,"%d",&m);
fscanf(infile,"%d",&n);
fscanf(infile,"%d",&facs);
fscanf(infile,"%d",&orthog);
fscanf(infile,"%d",&choice);
fscanf(infile,"%d",&level);
fscanf(infile,"%d",&creps);
fscanf(infile,"%d",&reps);
fscanf(infile,"%d",&c);
fscanf(infile,"%d\n",&row);

for (i=1; i<=n; i++)
  fscanf(infile,"%g",&var_label[i]);
fscanf(infile,"\n");

for (i=1; i<=m; i++) {
  for (j=1; j<=n; j++)
    fscanf(infile,"%g",&x[i][j]);
  fscanf(infile,"\n");
}

fclose(infile);

/* READ FROM FILE .FAC */
infile = fopen("exp.fac","r");

fscanf(infile,"%d",&fac_factors);
for (i=1; i<=factors; i++) {
  fscanf(infile,"%f %c %f",&low[i],&ch,&high[i]);
  printf(" %.2f %.2f\n",low[i],high[i]); }

fclose(infile);

/*  IF IT IS A 3-LEVEL DESIGN CREATE MID LEVEL & SEMIRANGE
VALUES */
if (level == 3) {
  for (j=1; j<=factors; j++)
    {
    mid[j]       = (high[j] + low[j])/2;
    semirange[j] = high[j] - mid[j];
    }
}

outfile = fopen("exp.raw","w");
```

```
if (gpscreen > 0) {

  /* READ FROM FILE .GRP */
  infile = fopen("exp.grp","r");

  for (i=1; i<=factors; i++) {
    fscanf(infile,"%d",&factor_groupnum[i]);
    printf("FACTOR # %d IS IN GROUP #
%d\n",i,factor_groupnum[i]); }

  fclose(infile);

  for (i=1; i<=factors; i++) {
    k = factor_groupnum[i];  /*  GIVES WHICH GROUP FACTOR IS
IN  */
    num_factors_ingroup[k]++;  /* COUNTER FOR THE # FACTORS
IN GROUP */
    l = num_factors_ingroup[k];
    /* PLACE THE FACTOR NUMBER IN THE GROUP NUMBER TABLE */
    group_table[l][k] = i;
  }

  for (i=1; i<=m; i++) { /* FOR ALL OF THE RUNS */
    printf("%2d ",i);
    fprintf(outfile,"%2d ",i);
    for (j=2; j<=(gpscreen+1); j++) { /* FOR ALL OF THE
FACTORS */
      for (k=1; k<=num_factors_ingroup[j-1]; k++) {
     factor_num = group_table[k][j-1];
     if (x[i][j] == -1) {           /* GROUP USES LOW VALUE */
       printf("%.2f ",low[factor_num]);
       fprintf(outfile,"%.2f ",low[factor_num]);
     }
     else {                         /* GROUP USES HIGH VALUE */
       printf("%.2f ",high[factor_num]);
       fprintf(outfile,"%.2f ",high[factor_num]);
     }
      }
      printf("    "); /* PUT SPACES BETWEEN THE GROUPS */
      fprintf(outfile,"    "); /* PUT SPACES BETWEEN THE
GROUPS */
    }
    printf("\n"); /* ONLY ONE RUN PER OUTPUT LINE */
    fprintf(outfile,"\n"); /* ONLY ONE RUN PER OUTPUT LINE
*/
  }
}
else {   /*  OTHER THAN GROUP RAW DATA CREATION  */
  for (i=1; i<=m; i++) { /* FOR ALL OF THE ROWS */
```

```
    printf("%2d ",i);
    fprintf(outfile,"%2d ",i);
    for (j=2; j<=(factors+1); j++) { /* FOR ALL OF THE
FACTORS */
      if (x[i][j] == -1) {        /* LOW VALUE */
     printf("%f ",low[j-1]);
     fprintf(outfile,"%f ",low[j-1]);
      }
      else if (x[i][j] == 1) {    /* HIGH VALUE */
     printf("%f ",high[j-1]);
     fprintf(outfile,"%f ",high[j-1]);
      }
      else if (x[i][j] == 0) {    /* MIDDLE VALUE */
     printf("%f ",mid[j-1]);
     fprintf(outfile,"%f ",mid[j-1]);
      }
      else {                      /* ALPHA CCD VALUE */
     printf("%f ",(mid[j-1]+(semirange[j-1]*x[i][j])));
     fprintf(outfile,"%f
",(mid[j-1]+(semirange[j-1]*x[i][j])));
      }  /* END OF IF STRUCTURE */
     }
    printf("\n");
    fprintf(outfile,"\n"); /* ONLY ONE RUN PER OUTPUT LINE
*/
  }
}
close(outfile);
printf("******************** PRESS RETURN TO CONTINUE
*******************\n");
ch = getchar();
} /*  END OF PROGRAM MAIN  */
```

REGPROC.PRG

```
*
* REGPROC.PRG
*
PROCEDURE p_varsel
PARAMETERS des_filename
  SAVE SCREEN
  CLEAR
  cvarsel(des_filename)  && reads des_filename makes
selection and
  RESTORE SCREEN         && writes exp.des as current file
  * get the file name of the save file
  overwrite = .F.
  prompt_msg = "INPUT THE DESIGN FILE NAME TO SAVE SELECTION
RESULTS"
```

```
  filename = input_filename(".DES",@overwrite,prompt_msg)
  IF !EMPTY(filename) .AND. overwrite
    COPY FILE exp.des TO &filename
  ENDIF
RETURN

PROCEDURE p_response
PUBLIC res_filename
IF FILE("exp.res")
  ERASE exp.res
ENDIF
* THIS ENSURES THAT ONLY EXP.RES IS THE CURRENT FILE FOR
REGRESSION
IF FILE("trans.res")
  ERASE trans.res
ENDIF
respond = .T.
@ 24,0 SAY SPACE(80)
@ 24,1 SAY "DO YOU WANT TO RETRIEVE *.RES FILE FROM DISK <Y>
?" ;
       GET respond PICT "Y"
READ
@ 24,0 SAY SPACE(80)
IF respond
  * read .RES file and create y_vector
  @ 0,0 SAY SPACE(80)
  @ 0,1 SAY "INPUT THE RESPONSE FILE NAME"
  res_filename = get_filename("*.RES")
  @ 0,0 SAY SPACE(80)
  IF !FILE(res_filename)
    @ 0,1 SAY "ERROR -> RESPONSE FILE DOES NOT EXIST
<RETURN>"
    wait_key = inkey(0)
    @ 0,0 SAY SPACE(80)
    RETURN
  ELSE
    * MAKE THE RETRIEVE FILE THE CURRENT RESPONSE FILE FOR
REGRESSION
    COPY FILE &res_filename TO exp.res
  ENDIF
ELSE
  @ 0,1 SAY "INPUT THE DESIGN FILE NAME THAT CORRENSPONDS TO
THE RESPONSES"
  des_filename = get_filename("*.DES")
  @ 0,0 SAY SPACE(80)
  IF !FILE(des_filename)
    @ 0,1 SAY "ERROR -> DESIGN FILE DOES NOT EXIST <RETURN>"
    wait_key = inkey(0)
    @ 0,0 SAY SPACE(80)
    RETURN
```

```
  ELSE
    DO p_flagval WITH des_filename
    num_runs = flag_val[3]
    PUBLIC y_vector[num_runs]
    afill(y_vector,0.000000)
    @ 0,0 SAY SPACE(80)
    @ 0,1 SAY "INPUT THE RESPONSES IN RUN ORDER"
    FOR i = 1 TO num_runs
      @ 24,1 SAY "FOR RUN # " + ltrim(str(i)) + ;
                 " ENTER RESPONSE VALUE => " ;
             GET y_vector[i]
      READ
      IF LASTKEY() = ESC
     @ 0,0 SAY SPACE(80)
     @ 24,0 SAY SPACE(80)
        RETURN
      ENDIF
    NEXT
    handle = FCREATE("EXP.RES")
    eol = chr(13) + chr(10)
    wbuffer = ltrim(str(num_runs)) + eol
    FWRITE(handle,wbuffer)
    wbuffer = ""
    FOR i = 1 TO num_runs
      wbuffer = ltrim(str(y_vector[i]))
      FWRITE(handle,wbuffer)
      IF i < num_runs
        FWRITE(handle,eol)
      ENDIF
      wbuffer = ""
    NEXT
    FCLOSE(handle)
    overwrite = .F.
    prompt_msg = "INPUT THE RESPONE FILE NAME TO SAVE
INPUTS"
    res_filename =
input_filename(".RES",@overwrite,prompt_msg)
    * IF NO SAVED FILENAME GIVEN THEN DO NOT COPY TO SAVE
.RES FILE
    IF !EMPTY(res_filename) .AND. overwrite
      COPY FILE exp.res TO &res_filename
    ENDIF
  ENDIF
ENDIF
RETURN

PROCEDURE p_transform
@ 0,1 SAY "TRANSFORMATIONS RESPONSE DATA USING THE FOLLOWING
FUNCTIONS"
exit_trans = .F.
```

```
save_trans_screen = savescreen(2,9,14,28)
prompt_size = 18
DO WHILE !exit_trans
  @ 2,9,14,28 BOX sl_box
  @ 24,0 SAY SPACE(80)
  @ 3,10  PROMPT menu_pad("Y**ALPHA",prompt_size);
          MESSAGE "SELECT FOR POWER TRANSFORMATION OF Y"
  @ 4,10  PROMPT menu_pad("LN(Y)",prompt_size);
          MESSAGE "SELECT FOR NATURAL LOG OF Y"
  @ 5,10  PROMPT menu_pad("LOG10(Y)",prompt_size);
          MESSAGE "SELECT FOR LOG BASE 10 OF Y"
  @ 6,10  PROMPT menu_pad("ARCSIN SQRT(Y)",prompt_size);
          MESSAGE "SELECT FOR ARCSIN SQUARE ROOT OF Y"
  @ 7,10  PROMPT menu_pad("LN( (1+Y)/(1-Y) )",prompt_size);
          MESSAGE "SELECT FOR NATURAL LOG OF (1+Y)/(1-Y)"
  @ 8,10  PROMPT menu_pad("(1/Y)",prompt_size);
          MESSAGE "SELECT FOR INVERSE OF Y"
  @ 9,10  PROMPT menu_pad("SQRT(Y)",prompt_size);
          MESSAGE "SELECT FOR THE SQUARE ROOT OF Y"
  @ 10,10 PROMPT menu_pad("Y**2",prompt_size);
          MESSAGE "SELECT FOR Y SQUARED"
  @ 11,10 PROMPT menu_pad("LN(B - Y)",prompt_size);
          MESSAGE "SELECT FOR NATURAL LOG OF (B - Y)"
  @ 12,10 PROMPT menu_pad("Original Values",prompt_size);
          MESSAGE "SELECT FOR THE ORIGNAL RESPONSE VALUES"
  @ 13,10 PROMPT menu_pad("Exit",prompt_size);
          MESSAGE "SELECT TO EXIT THIS MENU"
  choice = 1
  MENU TO choice
  t_value = 0.00000
  DO CASE
     CASE choice = 1
      @ 24,1 SAY "ENTER ALPHA VALUE FOR POWER TRANSFORMATION
=>" ;
             GET t_value
      READ
      @ 24,0 SAY SPACE(80)
     CASE choice = 9
      @ 24,1 SAY "ENTER B VALUE FOR NATURAL LOG
TRANSFORMATION =>" ;
             GET t_value
      READ
      @ 24,0 SAY SPACE(80)
  ENDCASE
  IF LASTKEY() != ESC
    IF choice = 11
      exit_trans = .T.
    ELSE
      IF choice = 10
        * make the current file exp.res instead of trans.res
```

```
          IF FILE("trans.res")
            ERASE trans.res
          ENDIF
          res_filename = "exp.res"
        ELSE
          res_filename = "trans.res"
          PRIVATE carray[3]
          carray[1] = "exp.res"
       carray[2] = choice
          carray[3] = t_value
          SAVE SCREEN
          CLEAR
          * call to C function to do transformations
          ctrans(carray)
          RESTORE SCREEN
        ENDIF
      ENDIF
    ENDIF
ENDDO
@ 0,0 SAY SPACE(80)
*  restore the screen for the facset screen
restscreen(2,9,14,28,save_trans_screen)
RETURN
```

<u>VARSEL.C</u>

```
/* VARSEL.C  TO REDUCE THE FACTOR SPACE FOR REGRESSION */
/*
*/

# include <string.h>
# include <stdio.h>
# include <math.h>
# include <dos.h>
# include "nandef.h"
# include "extend.h"

CLIPPER cvarsel()
{

FILE *infile, *outfile;
int i,j,k;
int gpscreen, factors, m, n;
int facs, orthog, choice, level, creps, reps, c, row;
int removed;
char ch;
```

```
char des_filename[12];

static float var_label[40];
static float x[40][40];
static int choose[40];

strcpy(des_filename,_parc(1,1));
if ((infile = fopen(des_filename, "r")) == NULL) {
  printf("File %s could not be opened\n", des_filename);
  return;
}

fscanf(infile,"%d",&gpscreen);
fscanf(infile,"%d",&factors);
fscanf(infile,"%d",&m);
fscanf(infile,"%d",&n);
fscanf(infile,"%d",&facs);
fscanf(infile,"%d",&orthog);
fscanf(infile,"%d",&choice);
fscanf(infile,"%d",&level);
fscanf(infile,"%d",&creps);
fscanf(infile,"%d",&reps);
fscanf(infile,"%d",&c);
fscanf(infile,"%d\n",&row);

printf("\n THE FOLLOWING IS THE FACTOR LIST FOR THE DESIGN
SELECTED\n");
for (i=1; i<=n; i++) {
  fscanf(infile,"%g",&var_label[i]);
  printf(" X%-.0f ",var_label[i]);
}
fscanf(infile,"\n");
printf("\n");

for (i=1; i<=m; i++) {
  for (j=1; j<=n; j++)
    fscanf(infile,"%g",&x[i][j]);
  fscanf(infile,"\n");
}

fclose(infile);

/* CASE TO SELECT VARIABLES */

removed = 0;
for (i=1; i<=n; i++) {
  printf("\nREMOVE VARIABLE => X%-.0f <= FROM THE
MODEL?\n",var_label[i]);
  printf("<1 = YES> TO REMOVE OR <RETURN> TO RETAIN
FACTOR\n");
```

```
  ch = getch();
  if (ch == '1') {
    removed = 1;
    choose[i] = 1;
    printf("\n***** FACTOR X%-.0f REMOVED FROM MODEL
*****\n",var_label[i]);
  }
}

if (removed == 0) {
  printf("NO VARIABLES SELECTED FOR REMOVAL <RETURN>\n");
  ch = getch();
  return;
}

/*  THIS LOOP ACTUALLY COMPRESSES THE MATRIX BASED ON THE
DELETIONS */
for (j=1; j<=n; j++) {
  if (choose[j] == 1) {
    for (k=j; k<=n-1; k++) {
      for (i=1; i<=m; i++) {
     x[i][k] = x[i][k+1]; }
      var_label[k] = var_label[k+1];
      choose[k] = choose[k+1];
    }
    n = n-1;
    j = j-1;
  }
}

printf("\nTHE X MATRIX\n");

for (i=1; i<=n; i++) printf("%6.0f ",var_label[i]);
printf("\n");
for (i=1; i<=m; i++) {
  for (j=1; j<=n; j++) {
    printf("%6.1f ",x[i][j]);
  }
  printf("\n");
}

printf("\n******************* PRESS RETURN
*******************");
getch();

/* WRITE exp.des AS CURRENT DESIGN FILE */
outfile  = fopen("exp.des", "w");

fprintf(outfile," %2d",gpscreen);
fprintf(outfile," %2d",factors);
```

```
fprintf(outfile," %2d",m);
fprintf(outfile," %2d",n);
fprintf(outfile," %2d",facs);
fprintf(outfile," %2d",orthog);
fprintf(outfile," %2d",choice);
fprintf(outfile," %2d",level);
fprintf(outfile," %2d",creps);
fprintf(outfile," %2d",reps);
fprintf(outfile," %2d",c);
fprintf(outfile," %2d\n",row);


for (i=1; i<=n; i++)
  fprintf(outfile,"%4g ",var_label[i]);
fprintf(outfile,"\n");

for (i=1; i<=m; i++) {
  for (j=1; j<=n; j++)
    fprintf(outfile,"%4g ",x[i][j]);
  fprintf(outfile,"\n");
}
fclose(outfile);

} /* END OF CVARSEL.C FUNCTION */
```

TFORM.C

```
/* THIS PROGRAM TRANSFORMS RESPONSE DATA */

# include <string.h>
# include <stdio.h>
# include <math.h>
# include "nandef.h"
# include "extend.h"
char transform_res_filename[12] = "trans.res";

CLIPPER ctrans()
{
FILE *infile, *outfile;
int i,j,k;
int choice,num_runs;
float min,alpha;
char res_filename[12];
float y[65],transform_y[65];
strcpy(res_filename,_parc(1,1));
```

```
choice = _parni(1,2);
alpha  = _parnd(1,3);
min = 1;

/* READ IN THE Y VECTOR FROM THE CORRECT FILE NAME */
infile = fopen(res_filename,"r");
fscanf(infile,"%d\n",&num_runs);
for (i=1; i<=num_runs; i++) fscanf(infile,"%g\n",&y[i]);
fclose(infile);

/* THE FUNCTION TRANSFORM IS USED FOR TRANSFORMATIONS ON
   THE RESPONSES (Y)  */

/* THE CALLING TERM: transform(y,m1,transf,alpha);  */
if (choice == 1)
  {
  /*  PICK A POWER TRANSFORMATION: Y**alpha  */
  for (i=1; i<=num_runs; i++)
      transform_y[i] = pow(y[i],alpha);
  }
else if (choice == 2)
  {
  /* NATURAL LOG  */
  for (i=1; i<=num_runs;  i++)
     if (y[i] < min) min = y[i];
  if (min < 0.) printf("TRANSFORMATION NOT POSSIBLE WITH
NEGATIVE VALUES\n");
  else
   {
    for (i=1; i<=num_runs; i++)
       if(y[i] != 0) transform_y[i] = log(y[i]);
   }
  }
else if (choice == 3)
  {
  /* LOG BASE 10  */
  for (i=1; i<=num_ uns;  i++)
    if (y[i] < min) min = y[i];
  if (min < 0.) printf("TRANSFORMATION NOT POSSIBLE WITH
NEGATIVE VALUES\n");
  else
   {
    for (i=1; i<=num_runs;  i++)
       if(y[i] != 0) transform_y[i] = log10(y[i]);
   }
  }
else if (choice == 4)
  {
  /* ARCSIN SQRT(Y) */
  for (i=1; i<=num_runs;  i++)
```

```
    if (y[i] < min) min = y[i];
  for (i=1; i<=num_runs;  i++)
    if (y[i] < -1. || y[i] > 1.) min = -1;
  if (min < 0.)
   {printf("TRANSFORMATION NOT POSSIBLE WITH NEGATIVE
VALUES\n");
    printf("AND VALUES MUST BE BETWEEN -1 AND +1 FOR ARCSIN
CALCULATION\n");
     }
  else
   {
    for (i=1; i<=num_runs;  i++)
     transform_y[i] = asin(sqrt(y[i]));
   }
  }
else if (choice == 5)
  {
  /* LOG [(1+Y)/(1-Y)] */
  for (i=1; i<=num_runs;  i++)
    if (y[i] < -1 || y[i] > 1) min = -1;
  if (min < 0.) printf("THE RESPONSES MUST BE BETWEEN 0 AND
1\n");
  else
   {
  for (i=1; i<=num_runs;  i++)
      transform_y[i] = log((1+y[i])/(1-y[i]));
   }
  }
else if (choice == 6)
  {
  /* INVERSE OF Y  */
  for (i=1; i<=num_runs;  i++)
    if(y[i] != 0) transform_y[i] = 1.0/y[i];
  }
else if (choice == 7)
  {
  /* SQUARE ROOT OF Y */
  for (i=1; i<=num_runs;  i++)
    if (y[i] < min) min = y[i];
  if (min < 0.) printf("TRANSFORMATION NOT POSSIBLE WITH
NEGATIVE VALUES\n");
  else
   {
    for (i=1; i<=num_runs;  i++)
     transform_y[i] = sqrt(y[i]);
   }
  }
else if (choice == 8)
  {
  /*  SQUARE OF Y  */
```

```
  for (i=1; i<=num_runs;  i++)
      transform_y[i] = y[i]*y[i];
  }
else if (choice == 9)
  {
  /* LN(B-Y) */
  for (i=1; i<=num_runs;  i++)
    if ((alpha - y[i]) < min) min = alpha - y[i];
  if (min <= 0.) printf("TRANSFORMATION NOT POSSIBLE WITH
NEGATIVE VALUES\n");
  else
   {
  for (i=1; i<=num_runs;  i++)
    if(alpha-y[i] != 0) transform_y[i] = log(alpha - y[i]);
   }
  }

printf("\nTHE ORIGINAL Y AND TRANSFORMED Y VECTOR(s)\n");
for (i=1; i<=num_runs; i++)
  printf("%10.5f             %10.5f\n",y[i],transform_y[i]);
printf("\n******************** PRESS RETURN
************************\n");
getch();

/* WRITE THE TRANSFORMED RESPONSE VECTOR TO FILE */
outfile = fopen(transform_res_filename,"w");
fprintf(outfile,"%d\n",num_runs);
for (i=1; i<=num_runs; i++)
  fprintf(outfile,"%g\n",transform_y[i]);
fclose(outfile);
}
```

<u>BROWN.PRG</u>

```
FUNCTION abrowse
PARAMETERS
ar,num_elems,key_var,start_row,start_col,end_row,end_col
PRIVATE
num_disp_rows,floor,ceiling,key,highlight,width,cur_disp_row
s
IF num_elems = 0
  RETURN (0)
ENDIF
SET CURSOR OFF
@ start_row,start_col,end_row,end_col BOX SL_BOX
num_disp_rows = end_row - start_row - 1
width = end_col - start_col - 1
floor = 1
highlight =1
cur_disp_rows = afill_box(ar,start_row,start_col,end_row,;
```

```
                                    end_col,floor)
ceiling = cur_disp_rows + floor - 1
IF ceiling < num_elems
  SET COLOR TO /W
  @ end_row,start_col SAY chr(25)
  SET COLOR TO W/
ENDIF
SET COLOR TO /W
* Highlight active element
@ start_row + highlight,start_col + 1 SAY;
  pad(ar[floor + highlight - 1],width)
SET COLOR TO W/
key = inkey(0)

* key_var is either ENTER or ESC to activate the ENTER case
DO WHILE (key != ESC).AND.(key != key_var)
  * Remove highlight from active element
  @ start_row + highlight,start_col + 1 SAY;
    pad(ar[floor + highlight - 1],width)
  DO CASE
    CASE key = ENTER
      IF asterisk[floor + highlight - 1]  && if element is
active
        * deactivate the element
        asterisk[floor + highlight - 1] = .F.
        ar[floor + highlight - 1] = ;
                   STUFF(ar[floor + highlight - 1],1,2,"  ")
      ELSE
        * activate the element
        asterisk[floor + highlight - 1] = .T.
        ar[floor + highlight - 1] = ;
                   STUFF(ar[floor + highlight - 1],1,2,">>")
      ENDIF
    CASE key = UP_ARROW
      IF highlight > 1
        highlight = highlight - 1
      ELSE
        IF floor > 1
          floor = floor - 1
          scroll(start_row + 1,start_col + 1,;
                 end_row - 1, end_col - 1,-1)
          * This leaves highlight row empty
          cur_disp_rows = min(num_disp_rows,;
                              num_elems - floor + 1)
          ceiling = floor + cur_disp_rows - 1
        ENDIF
      ENDIF
    CASE key = DOWN_ARROW
      IF highlight < cur_disp_rows
        highlight = highlight + 1
```

```
    ELSE
      IF ceiling < num_elems
        floor = floor + 1
        scroll(start_row + 1,start_col + 1,;
               end_row - 1, end_col - 1,1)
        * This leaves highlight row empty
        cur_disp_rows = min(num_disp_rows,num_elems -
floor + 1)
        ceiling = floor + cur_disp_rows - 1
      ENDIF
    ENDIF
  CASE key = PG_UP
    IF floor > 1
      floor = max(floor - num_disp_rows,1)
      ceiling = afill_box(ar,start_row,start_col,end_row,;
                          end_col,floor)
      cur_disp_rows = ceiling - floor + 1
    ENDIF

  CASE key = PG_DOWN
    IF ceiling < num_elems
      floor = min(floor + num_disp_rows,num_elems)
      cur_disp_rows = afill_box(ar,start_row,start_col,;
                                end_row,end_col,floor)
      ceiling = cur_disp_rows + floor - 1
      highlight = min(highlight,cur_disp_rows)
    ENDIF

  CASE key = HOME
    highlight = 1
  CASE key = END_KEY
    hightlight = cur_disp_rows
  CASE key = CTRL_HOME
    highlight = 1
    floor = 1
    cur_disp_rows = afill_box(ar,start_row,start_col,;
                              end_row,end_col,floor)
    ceiling = cur_disp_rows + floor - 1
  CASE key = CTRL_END
    IF ceiling = num_elems
      highlight = cur_disp_rows
    ELSE
      floor = num_elems - num_disp_rows + 1
      cur_disp_rows = afill_box(ar,start_row,start_col,;
                                end_row,end_col,floor)
      ceiling = cur_disp_rows + floor - 1
      highlight = cur_disp_rows
    ENDIF
  ENDCASE
  * Highlight active element
```

```
  SET COLOR TO /W
  @ start_row + highlight,start_col + 1 ;
    SAY pad(ar[floor + highlight - 1],width)
  SET COLOR TO W/

  IF floor > 1
    SET COLOR TO /W
    @ start_row,start_col SAY chr(24)
    SET COLOR TO W/
  ELSE
    @ start_row,start_col SAY chr(218)
  ENDIF

  IF ceiling < num_elems
    SET COLOR TO /W
    @ start_row,start_col SAY chr(25)
    SET COLOR TO W/
  ELSE
    @ start_row,start_col SAY chr(192)
  ENDIF

  key = inkey(0)
ENDDO
SET CURSOR OFF
RETURN(IF(lastkey() = ESC,0,floor + highlight -1))

FUNCTION afill_box
PARAMETERS ar,start_row,start_col,end_row,end_col,floor
PRIVATE num_disp,num_rows,i,width

num_rows = end_row - start_row - 1
width = end_col - start_col - 1
num_disp = min(num_rows,num_elems - floor + 1)

FOR i = 1 TO num_disp
  @ start_row + i,start_col + 1 SAY pad(ar[floor + i -
1],width)
NEXT
FOR i = num_disp + 1 TO num_rows
  @ start_row + i,start_col + 1 SAY space(width)
NEXT

RETURN(num_disp)

FUNCTION pad
PARAMETERS str,width
IF len(str) > width
  str = substr(str,1,width)
ELSE
  str = str + space(width - len(str))
```

```
ENDIF
RETURN (str + space(width - len(str)))
```

CONSTS.PRG

```
FUNCTION init_consts

PUBLIC
SL_BOX,BOX1,BOX2,ESC,UP_ARROW,PG_UP,DOWN_ARROW,PG_DOWN,CTRL_
END
PUBLIC
END_KEY,HOME,CTRL_HOME,ENTER,F1,F2,F3,F4,F5,F6,F7,F8,F9,F10
PUBLIC LT_ARROW,RT_ARROW,CTRL_W,CTRL_Y,CTRL_T
PUBLIC AC_IDLE,AC_TOP,AC_BOT,AC_EXCEP,AC_NOITEM,AC_ABORT
PUBLIC AC_CONTINUE,BRIGHT,NORM,ENH,AC_GO_MATCH,AC_SELECT

* achoices modes
AC_IDLE   = 0
AC_TOP    = 1
AC_BOT    = 2
AC_EXCEP  = 3
AC_NOITEM = 4

* achoice return values
AC_ABORT    = 0
AC_SELECT   = 1
AC_CONTINUE = 2
AC_GO_MATCH = 3

* color values for BW screen
BRIGHT = "W+/"
NORM   = "W/"
ENH    = "/W"

* single line box
SL_BOX = chr(218) + chr(196) + chr(191) + chr(179) +
chr(217) + ;
         chr(196) + chr(192) + chr(179)
BOX1   = chr(213) + chr(205) + chr(184) + chr(179) +
chr(217) + ;
         chr(196) + chr(192) + chr(179)
BOX2   = chr(201) + chr(205) + chr(187) + chr(186) +
chr(188) + ;
         chr(205) + chr(200) + chr(186)

* some key values from INKEY
ESC        = 27
UP_ARROW   = 5
PG_UP      = 18
DOWN_ARROW = 24
PG_DOWN    = 3
CTRL_END   = 23
END_KEY    = 6
```

```
HOME        = 1
CTRL_HOME   = 29
ENTER       = 13
LT_ARROW    = 19
RT_ARROW    = 4
F1          = 28
F2          = -1
F3          = -2
F4          = -3
F5          = -4
F6          = -5
F7          = -6
F8          = -7
F9          = -8
F10         = -9
CTRL_W      = 23
CTRL_Y      = 25
CTRL_T      = 20

RETURN(.T.)
```

FUNC.PRG

```
FUNCTION p_flagval
PARAM des_filename
PUBLIC flag_val[12]
handle = FOPEN(des_filename)
current_string = freadline(handle)
FOR i = 1 to 11
  temp_string    = LTRIM(current_string)
  blank_pos      = AT(" ",temp_string)
  flag_val[i]    = VAL(LEFT(temp_string,blank_pos-1))
  current_string = SUBSTR(temp_string,blank_pos)
NEXT
FCLOSE(handle)
flag_val[12]    = VAL(current_string)
RETURN(.T.)

FUNCTION input_filename
PARAMETERS extension_string,overwrite,prompt_msg
PRIVATE extension_string,filename,correct
  filename = SPACE(12)
  respond = .T.
  @ 24,0 SAY SPACE(80)
  @ 24,1 SAY "DO YOU TO SAVE " + extension_string + ;
            " FILE TO DISK <Y> ?" GET respond PICT "Y"
  READ
  @ 24,0 SAY SPACE(80)
  IF respond
    @ 0,1 SAY prompt_msg
    correct = .F.
    DO WHILE .NOT.correct  && loop to get the filename to
save
      filename = SPACE(8)
      @ 24,1 SAY "Input the  " + extension_string + ;
                "  file name (up to 8 chars) => " ;
             GET filename PICT "@! ANNNNNNN"
      READ
      IF LASTKEY() = ESC
     @ 0,0 SAY SPACE(80)
     RETURN(SPACE(12))
      ENDIF
      filename = ALLTRIM(filename) + extension_string
      @ 24,0 SAY SPACE(80)
      say_string = "Is the filename " + filename + " Correct
- <Y>"
      @ 24,1 SAY say_string GET correct PICT "Y"
      READ
      IF LASTKEY() = ESC
     @ 0,0 SAY SPACE(80)
        RETURN(SPACE(12))
```

```
      ENDIF
      @ 24,0 SAY SPACE(80)
      IF correct
        * CHECK TO SEE IF FILE EXISTS ON DISK
        IF FILE(filename)
          @ 24,1 SAY "FILE =>" + filename + "<= EXISTS -- DO
YOU WANT " +;
                     "TO OVERWRITE <N> ?" GET overwrite PICT
"Y"
          READ
          IF LASTKEY() = ESC
         @ 0,0 SAY SPACE(80)
            RETURN(SPACE(12))
          ENDIF
          @ 24,0 SAY SPACE(80)
          IF !overwrite
            correct = .F.     && ASK FOR ANOTHER FILE NAME
          ENDIF
        ELSE
          overwrite = .T.     && SEND BACK POSITIVE WRITE
MESSAGE
        ENDIF
      ENDIF
    ENDDO
  @ 0,0 SAY SPACE(80)
  ENDIF
RETURN(filename)


FUNCTION get_filename
PARAMETERS dir_search_string
PRIVATE dir_search_string,file_box[5],num_files,filename

      file_box[1] = "enter_title(sysparam)"
      file_box[2] = "rl_getfil(sysparam)"
      file_box[3] = "ok_button(sysparam)"
      file_box[4] = "cancel_button(sysparam)"
      file_box[5] = "filelist(sysparam)"

      filename = SPACE(12)
      num_files = adir(dir_search_string)
      state = 0
      IF num_files > 0
        PRIVATE files[num_files]
        adir(dir_search_string,files)
        okee_dokee = "do_it()"
        state = multibox(7,17,7,5,file_box)
      ENDIF

RETURN(IF(state = 0,SPACE(12),trim(substr(filename,1,12)) ))
```

```
FUNCTION freadln
PARAMETERS handle,buffer,max_line
PRIVATE line,BUF_SIZE,eol,num_read,SEEK_BOF,save_pos

* seek mode,absolute position
SEEK_BOF = 0
line = SPACE(max_line)
buffer = ""

* save current file position for later seek
save_pos = ftell(handle)
num_read = FREAD(handle,@line,max_line)
eol = AT(chr(13),substr(line,1,num_read))
IF eol = 0
  buffer = line
ELSE
  buffer = substr(line,1,eol-1)  && copy up to eol
  * now position file to next line (skip if) ...
  FSEEK(handle,save_pos + eol + 1,SEEK_BOF)
ENDIF
RETURN num_read != 0

FUNCTION freadline
PARAMETERS handle
ch = " "
buffer = ""
line_size = 0
num_read = fread(handle,@ch,1)
DO WHILE num_read = 1.AND.ch != chr(13)
  buffer = buffer + ch
  line_size = line_size + 1
  num_read = fread(handle,@ch,1)
ENDDO
RETURN buffer

FUNCTION menu_pad
PARAMETERS string,size
RETURN(string + space(size - len(string)))

FUNCTION ftell
PARAMETERS handle
RETURN FSEEK(handle,0,1)
```

```
     RLDIALOG.PRG
***
*    multibox()
*
*    sysparam values:
*         1    =    initialize and display
*         2    =    hilite (become the current item)
*         3    =    dehilite (become a non-current item)
*         4    =    become a selected item and return a new
state
*
*         note that the above values are interpreted
*               differently by each function
*
*    states:
*         0    =    abort the process
*         1    =    initialization
*         2    =    pointing (cursor)
*         3    =    entry/selection
*         4    =    complete the process
***
FUNCTION multibox

PARAMETERS wt, wl, wh, beg_curs, boxarray
PRIVATE funcn, sysparam, state, cursor, x
PRIVATE asel, arel, frame, lframe

asel       = 1
arel       = 0
frame      = "╒═╕│╛═╘│"
lframe     = "╤═╤│╛═╧│"

DECLARE box_row[LEN(boxarray)]
DECLARE box_col[LEN(boxarray)]

SAVE SCREEN

@ wt, wl, wt + wh + 1, wl + 45 BOX frame + " "

* state 1 ... initialization
sysparam = 1

FOR cursor = 1 TO LEN(boxarray)
     funcn = boxarray[cursor]
     x = &funcn

     * each function leaves the cursor at its top left
corner
     box_row[cursor] = ROW()
```

```
     box_col[cursor] = COL()

NEXT

cursor = beg_curs
state = 2

DO WHILE state <> 0 .AND. state <> 4
     * till completed or aborted
     funcn = boxarray[cursor]

     DO CASE

          CASE state = 2
               * pointing state
               sysparam = 2
               x = &funcn

               k = INKEY(0)

               DO CASE

                    CASE k = 13 .OR. jisdata(k)
                         * change to selection state
                         state = 3

                    CASE k = 27
                         * abort
                         state = 0

                    OTHERWISE
                         * current item becomes uncurrent
                         sysparam = 3
                         x = &funcn

                         * get a new cursor
                         cursor = matrix(cursor, k)

               ENDCASE

          CASE state = 3
               * be selected and return a new state
               sysparam = 4
               state = &funcn

     ENDCASE
ENDDO

RESTORE SCREEN
```

```
RETURN state


***
*    title
***
FUNCTION enter_title
PARAMETERS sysparam

IF sysparam = 1
     @ wt + 1, wl + 2 SAY "Enter a filename "

     * set cursor for initialization
     @ wt + 1, wl + 2 SAY ""

ENDIF

RETURN 2


FUNCTION save_title
PARAMETERS sysparam

IF sysparam = 1
     * watch out for the length of file, it may exceed the
box width (path)
     @ wt + 3, wl + 4 SAY "Save Changes To File " +
TRIM(filename) + "?"

     * set cursor for initialization
     @ wt + 3, wl + 4 SAY ""

ENDIF

RETURN 2

***
*    get filename
***
FUNCTION rl_getfil
PARAMETERS sysparam

DO CASE

     CASE sysparam = 1 .OR. sysparam = 3
          @ wt + 3, wl + 2 SAY "File " + SUBSTR(filename, 1,
20)

          IF sysparam = 1
               * set cursor for initialization
```

```
          @ wt + 3, wl + 9 SAY ""
     ENDIF

  CASE sysparam = 2
     * be current...hilite
     SET COLOR TO I
     @ wt + 3, wl + 7 SAY SUBSTR(filename, 1, 20)
     SET COLOR TO

  CASE sysparam = 4
     * be selected...perform a GET on entry field

     Note: any other 'isdata' key will also execute
selection
     IF k <> 13
          KEYBOARD CHR(k)
     ENDIF

     filename = jenter_rc(filename, wt + 3, wl + 7, 64,
"@K!S20")

     SET CURSOR ON
     READ
     SET CURSOR OFF

     IF LASTKEY() = 13 .AND. .NOT. EMPTY(filename)
          * filename has been selected...go to the ok
button
          filename = JPAD(filename,20)
          @ wt + 3, wl + 7 SAY filename
          to_ok()
     ENDIF
ENDCASE

RETURN 2

***
*    file list
***
FUNCTION filelist
PARAMETERS sysparam
PRIVATE c

DO CASE

  CASE sysparam = 1
     * clear the window
     scroll(wt + 1, wl + 31, wt + wh, wl + 44, 0)
     @ wt, wl + 30, wt + wh + 1, wl + 45 BOX lframe
```

```
          IF .NOT. EMPTY(files[1])
               * display the files list
               KEYBOARD CHR(27)

achoice(wt+1,wl+32,wt+wh,wl+43,files,"ch_func",0,asel,arel)

          ENDIF

          * set cursor for initialization
          @ wt + 1, wl + 32 SAY ""

     CASE sysparam = 2

          IF EMPTY(files[1])
               * cannot cursor onto an empty list
               KEYBOARD CHR(219)

          ELSE
               * set initial relative row and array element
               asel = asel - arel + ROW() - wt - 1
               arel = ROW() - wt - 1

               * do the list selection
               c =
achoice(wt+1,wl+32,wt+wh,wl+43,files,"ch_func",0,asel,arel)

               IF LASTKEY() = 13
                    * filename selected from list...set
memvar
                    filename = SUBSTR(files[c] + SPACE(64),
1, 64)

                    * display filename in entry blank
                    rl_getfil(3)

                    * go directly to ok button
                    to_ok()

               ELSE

                    IF LASTKEY() = 19
                         * the system responds to CHR(19) as
^S
                         KEYBOARD CHR(219)

                    ELSE
                         * send character to multibox
                         KEYBOARD CHR(LASTKEY())

                    ENDIF
```

```
                    ENDIF
          ENDIF
ENDCASE

RETURN 2


***
*    ok button
***
FUNCTION ok_button

PARAMETERS sysparam
PRIVATE ok, reply

ok = " Ok "
reply = 2

DO CASE

     CASE sysparam = 1 .OR. sysparam = 3
          @ wt + wh, wl + 9 SAY ok

          IF sysparam = 1
               * set cursor for initialization
               @ wt + wh, wl + 9 SAY ""

          ENDIF

     CASE sysparam = 2
          * be current...hilite
          SET COLOR TO I
          @ wt + wh, wl + 9 SAY ok
          SET COLOR TO

     CASE sysparam = 4

          IF &okee_dokee
               * a job well done...complete the process
               reply = 4
          ENDIF

ENDCASE

RETURN reply


***
*    cancel button
***
```

```
FUNCTION cancel_button

PARAMETERS sysparam
PRIVATE can, reply

can = " Cancel "
reply = 2

DO CASE

     CASE sysparam = 1 .OR. sysparam = 3
          @ wt + wh, wl + 17 SAY can

          IF sysparam = 1
               * set cursor for initialization
               @ wt + wh, wl + 17 SAY ""

          ENDIF

     CASE sysparam = 2
          * be current...hilite
          SET COLOR TO I
          @ wt + wh, wl + 17 SAY can
          SET COLOR TO

     CASE sysparam = 4
          * cancel selected...abort the process
          reply = 0

ENDCASE

RETURN reply

***
*    cancel button for save file box
***
FUNCTION can_button

PARAMETERS sysparam
PRIVATE can, reply

can = " Cancel "
reply = 2

DO CASE

     CASE sysparam = 1 .OR. sysparam = 3
          @ wt + wh, wl + 25 SAY can

          IF sysparam = 1
```

```
          * set cursor for initialization
          @ wt + wh, wl + 25 SAY ""

     ENDIF

CASE sysparam = 2
     * be current...hilite
     SET COLOR TO I
     @ wt + wh, wl + 25 SAY can
     SET COLOR TO

CASE sysparam = 4
     * cancel selected...abort the process
     reply = 0

ENDCASE

RETURN reply

***
*    no button
***
FUNCTION no_button

PARAMETERS sysparam
PRIVATE no, reply

no = " No "
reply = 2

DO CASE

     CASE sysparam = 1 .OR. sysparam = 3
          @ wt + wh, wl + 13 SAY no

          IF sysparam = 1
               * set cursor for initialization
               @ wt + wh, wl + 13 SAY ""

          ENDIF

     CASE sysparam = 2
          * be current...hilite
          SET COLOR TO I
          @ wt + wh, wl + 13 SAY no
          SET COLOR TO

     CASE sysparam = 4
          * 'No' selected...abort the process
          reply = 0
```

```
          no_save_flag = .T.
ENDCASE

RETURN reply


***
*    achoice user function
***
FUNCTION ch_func

PARAMETERS amod, sel, rel
PRIVATE k, r, srow, scol

srow = ROW()
scol = COL()

asel = sel
arel = rel
r = 2

IF asel > arel + 1
     * more files off screen up
     @ wt + 1, wl + 44 SAY CHR(24)

ELSE
     @ wt + 1, wl + 44 SAY " "

ENDIF

IF LEN(files) - asel > wh - 1 - arel
     * more files off screen down
     @ wt + wh, wl + 44 SAY CHR(25)

ELSE
     @ wt + wh, wl + 44 SAY " "

ENDIF

IF amod = 3
     k = LASTKEY()

     DO CASE

          CASE k = 27
               * escape key
               r = 0

          CASE k = 13 .OR. k = 19 .OR. k = 219
               * return or left arrow
```

```
                r = 1

           CASE k = 1
                * home key..top of list
                KEYBOARD CHR(31)

           CASE k = 6
                * end key..end of list
                KEYBOARD CHR(30)

      ENDCASE
ENDIF

@ M->srow, M->scol SAY ""
RETURN r


***
*    do_it()
*
*    called from the "Ok" button as "&okee_dokee"
*    this function normally completes the process
***
FUNCTION do_it

PRIVATE done, error_str

DO CASE

     * error if empty filename
     CASE EMPTY(filename)     && error, empty filename
          KEYBOARD CHR(5)
          done = .F.

     OTHERWISE
          done = .T.

ENDCASE

RETURN done


***
*    relocate cursor
***
FUNCTION matrix

PARAMETERS old_curs, k
PRIVATE old_row, old_col, test_curs, new_curs
```

```
old_row = ROW()
old_col = box_col[old_curs]
new_curs = old_curs
test_curs = old_curs

DO CASE

     CASE k = 19 .OR. k = 219
          * left arrow

          DO WHILE test_curs > 2
               test_curs = test_curs - 1

               IF box_col[test_curs] < old_col .AND.
box_row[test_curs] >= old_row

                    IF box_row[test_curs] <
box_row[new_curs] .OR. new_curs = old_curs
                         * best so far
                         new_curs = test_curs

                    ENDIF
               ENDIF
          ENDDO

     CASE k = 4
          * right arrow

          DO WHILE test_curs < LEN(box_col)
               test_curs = test_curs + 1

               IF box_col[test_curs] > old_col .AND.
box_row[test_curs] <= old_row

                    IF box_row[test_curs] >
box_row[new_curs] .OR. new_curs = old_curs
                         * best so far
                         new_curs = test_curs

                    ENDIF
               ENDIF
          ENDDO

     CASE k = 5
          * up arrow

          DO WHILE test_curs > 2
               test_curs = test_curs - 1
```

```
               IF box_row[test_curs] < old_row .AND.
box_col[test_curs] <= old_col

                    IF box_col[test_curs] >
box_col[new_curs] .OR. new_curs = old_curs
                         * best so far
                         new_curs = test_curs

                    ENDIF
               ENDIF
          ENDDO

     CASE k = 24
          * down arrow

          DO WHILE test_curs < LEN(box_row)
               test_curs = test_curs + 1

               IF box_row[test_curs] > old_row .AND.
box_col[test_curs] >= old_col

                    IF box_col[test_curs] <
box_col[new_curs] .OR. new_curs = old_curs
                         * best so far
                         new_curs = test_curs

                    ENDIF
               ENDIF
          ENDDO
ENDCASE

RETURN new_curs


***
*    go directly to ok button
***
FUNCTION to_ok

cursor = ascan(boxarray, "ok_button(sysparam)")
KEYBOARD CHR(219)
RETURN 0

******
*    jisdata()
*
*    determine if a key is data suitable for entry in place
******
FUNCTION jisdata
```

```
PARAMETERS k

RETURN (M->k >= 32 .AND. M->k < 249 .AND. M->k <> 219 .AND.
CHR(M->k) <> ";")


******
*    jenter_rc(r,c,max_len,pfunc)
*
*    entry in place
******
FUNCTION jenter_rc

PARAMETERS org_str,r,c,max_len,pfunc
PRIVATE wk_str, keystroke

wk_str = JPAD(org_str, max_len)
SET CURSOR ON

IF .NOT. EMPTY(pfunc)
     @ r,c GET wk_str PICTURE pfunc
ELSE
     @ r,c GET wk_str
ENDIF

READ
SET CURSOR OFF

keystroke = LASTKEY()

IF keystroke = 27
     wk_str = ""
ENDIF

RETURN (TRIM(wk_str))


******
*    jpad()
*
*    syntax: jpad( <expC>, <expN> )
*
*    return: <expC> padded with spaces so that len( <expC> )
= <expN>
******
FUNCTION jpad

PARAMETERS s, n

RETURN(SUBSTR(s + SPACE(n), 1, n))
```

REGRESS.EXE

Regress.exe links four object files, REGRESS1 (main file), REGRESS2, REGRESS3, and REGRESS4.

REGRESS1

```
/* REGRESS1.C PERFORMS LINEAR REGRESSION WITH LEAST SQUARES */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/* ZTBL IS A TABLE LOOKUP FOR THE RANKITS FUNCTION */
float ztbl[101] =
{
-2.500,-2.330,-2.055,-1.880,-1.750,-1.645,-1.555,-1.476,-1.405,-1
.340,
  -1.282,-1.226,-1.175,-1.126,-1.080,-1.037, -.995, -.954, -.915,
-.878,
   -.842, -.807, -.773, -.738, -.707, -.675, -.643, -.613, -.583,
-.554,
   -.525, -.496, -.468, -.440, -.412, -.385, -.358, -.332, -.305,
-.280,
   -.253, -.227, -.202, -.176, -.150, -.126, -.100, -.075, -.050,
-.025,
    .000,  .025,  .050,  .075,  .100,  .126,  .150,  .176,  .202,
.227,
    .253,  .280,  .305,  .332,  .358,  .385,  .412,  .440,  .468,
.496,
    .525,  .554,  .583,  .613,  .643,  .675,  .707,  .738,  .773,
.807,
    .842,  .878,  .915,  .954,  .995, 1.037, 1.080, 1.126, 1.175,
1.226,
   1.282, 1.340, 1.405, 1.476, 1.555, 1.645, 1.750, 1.880, 2.055,
2.330,
   2.500};

/* MAIN CONTROLS THE REGRESSION PROGRAM  */

main ()
{

FILE *infile, *outfile;
int i,j,k,l,m1,n1,n2,nn,p,q,r;      /*    COUNTERS       */
int m, n;                           /* # ROWS & COLUMNS   */
int totdf, regdf, errdf;            /* DEGREES OF FREEDOM */
int sstocdf, ssrcdf, ssec‹f;
```

```
float ssr, ssto, sse;                    /* SUM OF SQUARES  */
float ssrc, sstoc, ssec;
float msr, msto, mse;
float msrc, msec;
float freg, fregp, rsq, rsqa;
float sum, ybar, mm;

static float x[35][35];                  /* DESIGN MATRIX    */
static float y[35][2];                   /* RESPONSE VECTOR  */
static float beta[35][2];                /* BETA ESTIMATES   */
float betap[2][35];                      /* TRANSPOSED BETA  */
float ss[2][2];                          /* SS scratch pad   */
static float xpxinv[35][35];             /* INV(X'X)         */

float f, t, alph, alpha;                 /* CCD INV(X'X) CALCS */

static float stats[35][10];              /* REGRESSION STATISTICS  */
static float xtrassr[35][2];             /* EXTRA SS ss */
static float ssrftest[35];

double fvalue();                 /* F TEST P-VALUE  */
double tvalue();                 /* STUDENT T TEST P-VALUE  */

                                 /* LACK OF FIT VARIABLES  */
float sspe, mspe;
float sslf, mslf;
float flof, flofp;               /* F TEST, P-VALUE  */
int level, row, row1;
int facs, totrow;
int reps, creps;               /* # OF CENTER POINTS/REPS */
float reps1, creps1;           /* CONVERT TO FLOAT FOR CALS */
int c, sspedf, sslfdf;
float ybart[35];

int gpscreen;
int factors;
int orthog;                         /* ORTHOGONAL DESIGN = 1 */
int choice;                         /* 2 LEVEL: RESOLUTION  */
                                    /* 3 LEVEL: FULL=1, CCD=2,
BOX-BEHNKEN=3 */
char ch= ',';                       /* VARIOUS USES  */
char outname[24],output[10];  /* NAMES OF INPUT AND OUTPUT FILES
*/


/* READ FROM FILE EXP.DES */

if ((infile  = fopen("exp.des", "r")) != NULL)
   {
fscanf(infile,"%d",&gpscreen);
```

```
fscanf(infile,"%d",&factors);
fscanf(infile,"%d",&m);
fscanf(infile,"%d",&n);
fscanf(infile,"%d",&facs);
fscanf(infile,"%d",&orthog);
fscanf(infile,"%d",&choice);
fscanf(infile,"%d",&level);
fscanf(infile,"%d",&creps);
fscanf(infile,"%d",&reps);
fscanf(infile,"%d",&c);
fscanf(infile,"%d\n",&row);

for (i=1; i<=n; i++)
  fscanf(infile,"%g",&stats[i][5]);

for (i=1; i<=m; i++) {
  for (j=1; j<=n; j++)
    fscanf(infile,"%g",&x[i][j]);
  fscanf(infile,"\n");
}

fclose(infile);


/* READ RESPONSES FROM FILE */

if ( (infile = fopen("trans.res", "r")) ||
      (infile = fopen("exp.res", "r"))    != NULL)
    {
fscanf(infile,"%d\n",&m1);

for (i=1; i<=m1; i++)
  fscanf(infile,"%g",&y[i][1]);

fclose(infile);

/* BEGIN REGRESSION PROGRAM */
/* IDIOT CHECKS AND PROGRAM LIMITS */

if (m1 != m)
  { printf("\nNEED THE SAME NUMBER OF RESPONSES AS ROWS IN DESIGN
MATRIX <RETURN>\n");
    getch();
    return; }
if (n > m)
  { printf("\nCAN'T PERFORM REGRESSION WITH MORE COLUMNS THAN
ROWS <RETURN>\n");
    getch();
    return; }
if (m >= 34)
```

```
   { printf("\nPROGRAM IS LIMITED TO 34 ROWS  <RETURN>\n");
     getch();
     return; }
if (n >=34)
   { printf("\nPROGRAM IS LIMITED TO 34 COLUMNS <RETURN>\n");
     getch();
     return; }
if (n < 2)
   { printf("\nMUST HAVE AT LEAST ONE VARIABLE IN MODEL
<RETURN>\n");
     getch();
     return; }
if (m < 4)
   { printf("\nMUST HAVE AT LEAST FOUR ROWS IN DESIGN MATRIX
<RETURN>\n");
     getch();
     return; }


creps = creps - 1;      /* ADJUST CEPS TO MATCH THIS PROGRAM */
creps1 = creps + 1.;      /* I.E., 1 REP = 2 CENTER POINTS */
reps1 = reps + 1.;        /* I.E., 1 REP = 2 DESIGNS       */

/* DEGREES OF FREEDOM */
ssrcdf = n-1;
ssecdf = m-n;
if (ssecdf == 0) ssecdf = 1;
sstocdf = m-1;

sspedf = m-c;
sslfdf = c-n;
totrow = m;          /* # OF ROWS IN DESIGN MATRIX  */


/* CALCULATE UNCORRECTED SUM OF SQUARES */

  /* CALCULATE INV(X'X)  */

  xpxinvf(x,xpxinv,m,n,orthog,level,facs,row,choice,creps,reps1);


  /* CALCULATE BETAS */

  betaf(xpxinv,x,y,beta,m,n);


  /* CALCULATE Y-HAT STATSISTICS  */

  yhatf(x,beta,stats,m,n);
```

```
  /* CALCULATE UNCORRECTED SSR  */

  ssrf(beta, x, y, m, n, ss);
  ssr = ss[1][1];


  /* CALCULATE UNCORRECTED SSTO  */
  sstof(y,ss,m);
  ssto = ss[1][1];


  /* CALCULATE UNCORRECTED SSE  */

  sse = ssto - ssr;
  if (sse <= .0001) sse = 1.;

  /* CALCULATE UNCORRECTED MEAN SQUARED ERRORS
  totdf = m;
  regdf = n;
  errdf = m-n;
  msr = ssr/regdf;
  msto = ssto/totdf;
  mse = sse/errdf;           */


/* CALCULATION FOR SS CORRECTION */

sum = 0;

for (i=1; i<=m; i++)
     sum = sum + y[i][1];


/* CALCULATE CORRECTED SUMS OF SQUARES  */

ssrc = ssr - (sum*sum)/m;

sstoc = ssto - (sum*sum)/m;

ssec = sse;

/* CALCULATE CORRECTED MEAN SQUARE ERRORS */

msrc = ssrc/ssrcdf;
msec = ssec/ssecdf;


/*  MODEL STATISTICS: F TEST, P-VALUE, R SQUARED, AND ADJ R
SQUARED  */
```

```
freg = msrc/msec;
fregp = fvalue(freg,ssrcdf,ssecdf);
rsq = ssrc/sstoc;
rsqa = 1. - (sstocdf/ssecdf)*(ssec/sstoc);


/* COEFFICIENT STATISTICS: STANDARD ERRORS (stats[i][1]),
   VARIANCES (stats[i][2]), T-STATISTICS (stats[i][3]),
   AND P-VALUES (stats[i][4]) OF THE COEFFICIENTS  */

for (i=1; i<=n; i++)
   { beta[i][0] = stats[i][5];
     stats[i][2] = xpxinv[i][i]*msec;
         stats[i][1] = sqrt(stats[i][2]);
     stats[i][3] = beta[i][1]/stats[i][1];
     stats[i][4] = tvalue(stats[i][3],ssecdf);
    }


/* SORT THE COEFFICIENTS ON P-VALUES */

for (i=1; i<=n-1; i++)
   { k = i;
     for (nn=0; nn<=5; nn++)
        stats[0][nn] = stats[i][nn];
     for (j = i+1; j<=n; j++)
        { if (stats[j][4] < stats[0][4])
            { k = j;
              for (nn=0; nn<=5; nn++)
                  stats[0][nn] = stats[j][nn];
            }
        }
     for (nn=0; nn<=5; nn++)
        { stats[k][nn] = stats[i][nn];
          stats[i][nn] = stats[0][nn];
        }
   }


/* CALCULATE EXTRA SUMS OF SQUARES */

nn = n;
for (i=1; i<=n; i++) xtrassr[i][0] = stats[i][5];

if (level == 2)
  {
   if (orthog == 1)  xby(x, beta, y, xtrassr, m, n);

   else
```

```
      {
       p = 1;
        if (n > 10) nn = 10,
       extrassr(x, beta, y, xtrassr, m, p, nn, sum);
      }
  }
else if (level == 3)
  {
   if (orthog == 1)  xby(x, beta, y, xtrassr, m, n);

   else if (orthog == 0 && (choice == 1 || choice == 3))
      {
       xby(x, beta, y, xtrassr, m, n);
       p = facs+2;
       nn = 2*facs + 1;
       extrassr(x, beta, y, xtrassr, m, p, nn, sum);
      }
   else if (orthog == 0 && choice == 2)
      {
       xby(x, beta, y, xtrassr, m, n);
       p = facs+2;
       nn = 2*facs + 1;
       extrassr(x, beta, y, xtrassr, m, p, nn, sum);
      }
   else
      {
       p = 1;
        if (n > 10) nn = 10;
       extrassr(x, beta, y, xtrassr, m, p, nn, sum);
      }
  }
else
   {
       p = 1;
        if (n > 10) nn = 10;
       extrassr(x, beta, y, xtrassr, m, p, nn, sum);
   }


/* SORT THE EXTRA INDIVIDUAL SS FROM LARGEST ON DOWN */
for (i=2; i<=n-1; i++)
   { k = i;
     for (nn=0; nn<=1; nn++)
          xtrassr[0][nn] = xtrassr[i][nn];
     for (j = i+1; j<=n; j++)
        { if (xtrassr[j][1] < xtrassr[0][1])
             { k = j;
               for (nn=0; nn<=1; nn++)
                    xtrassr[0][nn] = xtrassr[j][nn];
             }
```

```
        }
      for (nn=0; nn<=1; nn++)
        { xtrassr[k][nn] = xtrassr[i][nn];
          xtrassr[i][nn] = xtrassr[0][nn];
        }
   }

/* CALCULATE F TEST ON THE EXTRA SS */

for (i=1; i<=n; i++) ssrftest[i] = xtrassr[i][1]/msec;


/* CALCULATE RANKITS(stats[i][9])  */

rankits(stats,m,ztbl);



            /* CALCULATE LACK OF FIT   */

/* FOR TWO LEVEL DESIGNS WITH CENTER POINTS AND/OR REPS  */

for (i=1; i<=totrow; i++)  ybart[i] = 0.;
sspe = 0.;

if (creps >= 1 || reps >= 1)
{

if (level == 2)
  {
    if (creps < 1 && reps == 0)
      { printf("CANNOT COMPUTE LACK OF FIT WITHOUT REPS\n"); }

    else if (creps < 0 && reps >= 1)
      {
        /* 1 -1 ROWS  */
        row1 = pow(2,facs);
               k=1;
        while (k <= row1)
           {
             for (i=k; i<=totrow; i=i+row1)
                { ybart[k] = ybart[k] + y[i][1];
                }
             k = k+1;
           }

             for (k=1; k<=row1; k++)
                {  ybart[k] = ybart[k] / reps1;
                }
```

```
        k=1;
        while (k<=row1)
            {
              for (i=k; i<=totrow; i=i+row1)
                  {
                    sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
                  }
              k = k+1;
            }
    }

    else if (creps >= 1 && reps == 0)
      {
        /* CENTER POINTS */

        k = row + 1;
        for (i=(row+1); i<=(row+1+creps); i++)
            {
              ybart[k] = ybart[k] + y[i][1];
            }

        ybart[k] = ybart[k] / creps1;

        for (i=(row+1); i<=(row+1+creps); i++)
            {
              sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
            }

      }


    else if (creps >= 0 && reps >= 1)
      {
        /* 1 -1 ROWS   */
        row1 = pow(2,facs);
        k=1;
        while (k <= row1)
            {
              for (i=k; i<=row; i=i+row1)
                  {
                    ybart[k] = ybart[k] + y[i][1];
                  }
              k = k+1;
            }

              for (k=1; k<=row1; k++)
                  {  ybart[k] = ybart[k] / reps1;
                  }
```

```
        k=1;
        while (k<=row1)
            {
              for (i=k; i<=row; i=i+row1)
                  {
                    sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
                  }
              k = k+1;
            }

        /* CENTER POINTS  */

        k = row + 1;
        for (i=row+1; i<=row+(creps1*reps1); i++)
            {
              ybart[k] = ybart[k] + y[i][1];
            }

        ybart[k] = ybart[k] / (creps1*reps1);

        for (i=row+1; i<=row+(creps1*reps1); i++)
            {
              sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
            }
      }

   }


/* LACK OF FIT FOR CCD DESIGNS  */


if (level == 3  && choice == 2)    /*  choice = 2 = ccd design */
   {
     if (creps < 1 && reps == 0)
       { printf("CANNOT COMPUTE LACK OF FIT WITHOUT REPS\n"); }

     else if (creps < 1 && reps >= 1)
       {
         /* 1 -1 ROWS */
         row1 = pow(2,facs);
         k=1;
         while (k <= row1)
             {
               for (i=k; i<=row1*(reps+1); i=i+row1)
                   {
                     ybart[k] = ybart[k] + y[i][1];
```

```
                }
            k = k+1;
          }

            for (k=1; k<=row1; k++)
                {  ybart[k] = ybart[k] / reps1;
                }

      k=1;
      while (k<=row1)
          {
            for (i=k; i<=row1*(reps+1); i=i+row1)
                {
                  sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
                }
            k = k+1;
          }

      /* CENTER POINTS AND ALPHA ROWS */

     nn= (creps+1) + (2*facs);
      k = row + 1;
      while (k <= row + nn)
          {
            for (i=k; i<=totrow; i=i+nn)
                {
                  ybart[k] = ybart[k] + y[i][1];
                }
            k = k+1;
          }

            for (k=row+1; k<=row+nn; k++)
                {  ybart[k] = ybart[k] / reps1;
                }

      k=row+1;
      while (k<=row+nn)
          {
            for (i=k; i<=totrow; i=i+nn)
                {
                  sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
                }
            k = k+1;
          }

  }
```

```
else if (creps >= 1 && reps == 0)
  {
    /* CENTER POINTS */

    k = row + 1;
    for (i=(row+1); i<=(row+1+creps); i++)
        {
          ybart[k] = ybart[k] + y[i][1];
        }

    ybart[k] = ybart[k] / creps1;

    for (i=(row+1); i<=(row+1+creps); i++)
        {
          sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
        }

  }


else if (creps >= 1 && reps >= 1)
  {
    /* 1 -1 ROWS */
    row1 = pow(2,facs);
    k=1;
    while (k <= row1)
        {
          for (i=k; i<=row1*(reps+1); i=i+row1)
              {
                ybart[k] = ybart[k] + y[i][1];
              }
          k = k+1;
        }

          for (k=1; k<=row1; k++)
              {  ybart[k] = ybart[k] / reps1;
              }

    k=1;
    while (k<=row1)
        {
          for (i=k; i<=row1*(reps+1); i=i+row1)
              {
                sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
              }
          k = k+1;
        }
```

```
/* CENTER POINTS  */

nn= (creps+1) + (2*facs);
 k = row + 1;
 m1 = row+1;
 while (m1 <= totrow)
     {
       for (i=m1; i<=m1+creps; i++)
           {
             ybart[k] = ybart[k] + y[i][1];
           }
       m1 = m1+nn;
     }

 ybart[k] = ybart[k] / (reps1*creps1);

 m1 = row+1;
 while (m1 <= totrow)
     {
       for (i=m1; i<=m1+creps; i++)
       {
        sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
       }
       m1 = m1+nn;
     }

 /* ALPHA ROWS  */

 k = row + (creps+1) + 1;
 while (k <= row + nn)
     {
       for (i=k; i<=totrow; i=i+nn)
           {
             ybart[k] = ybart[k] + y[i][1];
           }
       k = k+1;
     }

 for (k=row+(creps+1)+1; k<=row+nn; k++)
     {  ybart[k] = ybart[k] / reps1;
     }

 k = row + (creps+1) + 1;
 while (k<=row + nn)
     {
       for (i=k; i<=totrow; i=i+nn)
           {
             sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
```

```
                }
              k = k+1;
            }
      }

   }

 /* THREE LEVEL DESIGNS  */

 if (level == 3 && (choice == 1 || choice == 3))
   {
     if (creps < 1 && reps == 0)
       { printf("CANNOT COMPUTE LACK OF FIT WITHOUT REPS\n"); }

     else if (creps < 0 && reps >= 1)
       {
         /* 1 -1 ROWS  */
         row1 = pow(3,facs);
         k=1;
         while (k <= row1)
            {
              for (i=k; i<=totrow; i=i+row1)
                 {
                   ybart[k] = ybart[k] + y[i][1];
                 }
              k = k+1;
            }

              for (k=1; k<=row1; k++)
                 {  ybart[k] = ybart[k] / reps1;
                 }

         k=1;
         while (k<=row1)
            {
              for (i=k; i<=totrow; i=i+row1)
                 {
                   sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
 ybart[k]);
                 }
              k = k+1;
            }
       }

     else if (creps >= 1 && reps == 0)
       {
         /* CENTER POINTS */

         k = row + 1;
         for (i=(row+1); i<=(row+1+creps); i++)
```

```
            {
              ybart[k] = ybart[k] + y[i][1];
            }

         ybart[k] = ybart[k] / creps1;

         for (i=(row+1); i<=(row+1+creps); i++)
            {
              sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
            }

       }

     else if (creps >= 0 && reps >= 1)
       {
         /* 1 -1 ROWS  */
         row1 = pow(3,facs);
         k=1;
         while (k <= row1)
            {
              for (i=k; i<=row; i=i+row1)
                 {
                   ybart[k] = ybart[k] + y[i][1];
                 }
              k = k+1;
            }

              for (k=1; k<=row1; k++)
                 {  ybart[k] = ybart[k] / reps1;
                 }

         k=1;
         while (k<=row1)
            {
              for (i=k; i<=row; i=i+row1)
                 {
                   sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
                 }
              k = k+1;
            }

         /* CENTER POINTS  */

         k = row + 1;
         for (i=row+1; i<=totrow; i++)
            {
              ybart[k] = ybart[k] + y[i][1];
```

```
            }

        ybart[k] = ybart[k] / (creps1*reps1);

        for (i=row+1; i<=totrow; i++)
            {
              sspe = sspe + (y[i][1] - ybart[k])*(y[i][1] -
ybart[k]);
            }
       }

  }

/* LACK OF FIT STATISTICS  */

mspe = sspe/sspedf;
sslf = ssec - sspe;
mslf = sslf/sslfdf;
if (mspe <= .000001) mspe = .0001;
flof = mslf/mspe;
flofp = fvalue(flof,sslfdf,sspedf);
}

outfile = fopen("regress.out","w");

fprintf(outfile,"%d\n",m);
fprintf(outfile,"%d\n",n);
fprintf(outfile,"%d\n",facs);
fprintf(outfile,"%d\n",orthog);
fprintf(outfile,"%d\n",choice);
fprintf(outfile,"%d\n",level);
fprintf(outfile,"%d\n",creps);
fprintf(outfile,"%d\n",reps);
fprintf(outfile,"%d\n",c);
fprintf(outfile,"%d\n",row);

fprintf(outfile,"%g\n",ssrc);
fprintf(outfile,"%d\n",ssrcdf);
fprintf(outfile,"%g\n",msrc);
fprintf(outfile,"%g\n",freg);
fprintf(outfile,"%g\n",ssec);
fprintf(outfile,"%d\n",ssecdf);
fprintf(outfile,"%g\n",msec);
fprintf(outfile,"%g\n",sstoc);
fprintf(outfile,"%d\n",sstocdf);
fprintf(outfile,"%g\n",fregp);
fprintf(outfile,"%g\n",rsq);
fprintf(outfile,"%g\n",rsqa);

fprintf(outfile,"%g\n",sslf);
```

```
fprintf(outfile,"%d\n",sslfdf);
fprintf(outfile,"%g\n",mslf);
fprintf(outfile,"%g\n",sspe);
fprintf(outfile,"%d\n",sspedf);
fprintf(outfile,"%g\n",mspe);
fprintf(outfile,"%g\n",flof);
fprintf(outfile,"%g\n",flofp);

for (i=1; i<=m; i++)
    for (j=1; j<=n; j++)
                  fprintf(outfile,"%g %c",x[i][j],ch);

fprintf(outfile,"\n");

for (i=1; i<=m; i++)
                  fprintf(outfile,"%g %c",y[i][1],ch);

fprintf(outfile,"\n");

for (i=1; i<=m; i++)
    for (j=0; j<=9; j++)
                  fprintf(outfile,"%g %c",stats[i][j],ch);

fprintf(outfile,"\n");

for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
                  fprintf(outfile,"%g %c",xpxinv[i][j],ch);

fprintf(outfile,"\n");

for (i=1; i<=m; i++)
    for (j=0; j<=1; j++)
                  fprintf(outfile,"%g %c",beta[i][j],ch);

fprintf(outfile,"\n");

for (i=1; i<=m; i++)
    for (j=0; j<=2; j++)
                  fprintf(outfile,"%g %c",xtrassr[i][j],ch);

fprintf(outfile,"\n");

for (i=1; i<=m; i++)
                  fprintf(outfile,"%g %c",ssrftest[i],ch);

fclose(outfile);

  }
   else
```

```
  { printf("\nCAN'T OPEN RESPONSE FILE <RETURN>\n");
    getch();
  }

 }
  else
 { printf("\nCAN'T OPEN DESIGN MATRIX FILE <RETURN>\n");
   getch();
 }

}
```

REGRESS2

```
/* FUNRES10.C SUPPORTS LINEAR REGRESSION WITH LEAST SQUARES */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/* CALCULATE INV(X'X)  */

xpxinvf(x,xpxinv,m,n,orthog,level,facs,row,choice,creps,reps1)
float x[35][35], xpxinv[35][35], reps1;
int m, n, orthog, level, facs, row, choice, creps;
{
float scratch[35][35], scratchy[35][35], alph, f, t;
int i, j;

transpos(x,scratch,m,n);

matmulmm(scratch,x,scratchy,n,m,n);

if (orthog == 1)
  {
   if (level == 2 && facs < 12)
     {xpxinv[1][1] = 1./m;
          for (i=2; i<=n; i++)
            xpxinv[i][i] = 1./row;
     }
   else if (level == 3 && choice == 1)
     invert(scratchy, xpxinv, n);
   else if (level == 3 && choice == 2)
     {
      alph = x[row+creps+1+2][2];
      f = row;
      t = (2*facs+creps+1)*(reps1);
      xpxinv[1][1] = 1./m;
      for (i=2; i<= facs+1; i++)
          xpxinv[i][i] = 1./(row + 2*alph*alph);
      for (i=facs+2; i<=2*facs+1; i++)
          xpxinv[i][i] = 1./((f*t - 4*f*alph*alph - 4*pow(alph,4)
+
                          2*(f+t)*pow(alph,4))/(f+t));
      for (i=2*facs+2; i<=n; i++)
          xpxinv[i][i] = 1./row;
     }
   else if (level == 3 && choice == 3)
     {
     invert(scratchy, xpxinv, n);
```

```
      }
   }
else if (orthog == 0) invert(scratchy, xpxinv, n);

}

/* CALCULATE BETAS */

betaf(xpxinv,x,y,beta,m,n)
float xpxinv[35][35], x[35][35], y[35][2], beta[35][2];
int m, n;

{
float scratch[35][35], scratchy[35][35];
int i;

transpos(x, scratch, m, n);

matmulmm(xpxinv, scratch, scratchy, n, n, m);

matmulm1(scratchy, y, beta, n, m, 1);

}

/* CALCULATE Y-HAT STATSISTICS  */

yhatf(x,beta,stats,m,n)
float x[35][35], beta[35][2], stats[35][10];
int m, n;

{
float scratch3[35][2];
int i;

matmulm1(x,beta,scratch3,m,n,1);

for (i=1; i<=m; i++)  stats[i][6] = scratch3[i][1];
for (i=1; i<=n; i++)  stats[i][0] = beta[i][1];

}

/* CALCULATE UNCORRECTED SSR  */

ssrf(beta, x, y, m, n, ss)
float beta[35][2], x[35][35], y[35][2], ss[2][2];
int m, n;
{

float betap[2][35], scratch[35][35], scratch1[2][35];
```

```
transpos2(beta, betap, n, 1);

transpos(x,scratch,m,n);

matmul1m(betap, scratch, scratch1, 1, n, m);

matmul11(scratch1, y, ss, 1, m, 1);

}

/* CALCULATE UNCORRECTED SSTO  */

sstof(y,ss,m)
float y [35][2], ss[2][2];
int m;
{
float scratch1[2][35];


transpos2(y, scratch1, m, 1);

matmul11(scratch1, y, ss, 1, m, 1);

}



/*  FUNCTION MATMULmm MULTIPLIES MATRIX a (mxn) BY MATRIX b (nxm)
    AND CREATES MATRIX c (mxm) */

matmulmm(float a[35][35], float b[35][35], float c[35][35],
       int m, int n, int p)


{
int i, j, k;
float sum;


   for (i=1; i<=m; i++)

      {for (j=1; j<=p; j++)

          {sum = 0.0;

           for (k=1; k<=n; k++)
              { sum = sum + a[i][k] * b[k][j]; }
```

```
              c[i][j] = sum;
             }
         }

}

/*   FUNCTION MATMULnn MULTIPLIES MATRIX a (nx1) BY MATRIX b (1xn)
     AND CREATES MATRIX c (nxn)   */

matmulnn(float a[35][2], float b[2][35], float c[35][35],
       int m, int n, int p)


{
int i, j, k;
float sum;


   for (i=1; i<=m; i++)

      {for (j=1; j<=p; j++)

          {sum = 0.0;

           for (k=1; k<=n; k++)
              { sum = sum + a[i][k] * b[k][j]; }

           c[i][j] = sum;
          }
      }

}

/*   FUNCTION MATMUL11 MULTIPLIES MATRIX a (1xn) BY MATRIX b (nx1)
     AND CREATES MATRIX c (1x1) */

matmul11(float a[2][35], float b[35][2], float c[2][2],
       int m, int n, int p)


{
int i, j, k;
float sum;


   for (i=1; i<=m; i++)

      {for (j=1; j<=p; j++)

          {sum = 0.0;
```

```
            for (k=1; k<=n; k++)
               { sum = sum + a[i][k] * b[k][j]; }

            c[i][j] = sum;
           }
       }

}
```

```
/*   FUNCTION MATMULm1 MULTIPLIES MATRIX a (mxn) BY MATRIX b (nx1)
     AND CREATES MATRIX c (mx1) */

matmulm1(float a[35][35], float b[35][2], float c[35][2],
       int m, int n, int p)


{
int i, j, k;
float sum;


   for (i=1; i<=m; i++)

      {for (j=1; j<=p; j++)

          {sum = 0.0;

           for (k=1; k<=n; k++)
              { sum = sum + a[i][k] * b[k][j]; }

           c[i][j] = sum;
          }
      }

}
```

```
/*   FUNCTION MATMUL1m MULTIPLIES MATRIX a (1xn) BY MATRIX b (nxm)
     AND CREATES MATRIX c (1xm) */

matmul1m(float a[2][35], float b[35][35], float c[2][35],
       int m, int n, int p)


{
int i, j, k;
float sum;
```

```
    for (i=1; i<=m; i++)

       {for (j=1; j<=p; j++)

           {sum = 0.0;

            for (k=1; k<=n; k++)
               { sum = sum + a[i][k] * b[k][j]; }

            c[i][j] = sum;
           }
       }

}


/* FUNCTION transpos COMPUTES THE TRANPOSE OF A MATRIX a (mxn)
   AND CREATES MATRIX ap (nxm)     */

transpos(float a[35][35], float ap[35][35], int m, int n)


{
int i,j;

   for (i=0; i<=m; i++)
       for (j=0; j<=n; j++)
           ap[j][i] = a[i][j];
}


/* FUNCTION transpos1 COMPUTES THE TRANPOSE OF A MATRIX a (1xm)
   AND CREATES MATRIX ap (mx1)     */

transpos1(float a[2][35], float ap[35][2], int m, int n)


{
int i,j;

   for (i=0; i<=m; i++)
       for (j=0; j<=n; j++)
           ap[j][i] = a[i][j];
}

/* FUNCTION transpos2 COMPUTES THE TRANPOSE OF A MATRIX a (mx1)
   AND CREATES MATRIX ap (1xm)     */

transpos2(float a[35][2], float ap[2][35], int m, int n)
```

```
{
int i,j;

   for (i=0; i<=m; i++)
       for (j=0; j<=n; j++)
           ap[j][i] = a[i][j];
}
```

REGRESS3

```
/* FUNRES11.C SUPPORTS LINEAR REGRESSION WITH LEAST SQUARES */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/* THIS FUNCTION CALCULATES THE RANKITS   */
/* CALLING TERM: rankits(stats,m); */


rankits(stats,m,ztbl)

float stats[35][10],ztbl[101];
int m;
{
float rtmse, blom, blom100, remain, zvalue, newz, blomz;
int i, intblom, intbloml;

/* rtmse = sqrt(msec); */

for (i=1; i<=m; i++)
   { blom = (i-.375)/(m+.25);
     blom100 = blom*100;
     intblom = floor(blom100);
     remain = blom100 - intblom;
     intbloml = intblom+1;
     zvalue = ztbl[intbloml] - ztbl[intblom];
     newz = zvalue*remain;
     blomz = ztbl[intblom] + newz;
     stats[i][9] = blomz;
   }

}

/*  FUNCTION GAMMLN IS USED TO CALCULATE P-VALUES   */

double gammln(xx)
double xx;
{

/*
*****************************************************************
****
This is an incomplete gamma function pulled from Numerical
Recipes.  It is
used to find T values and F values.
```

```
*****************************************************************
***  */

/* VAR */
double   x,tmp,ser;
int   j;
double   cof[6];
double stp, half, one, fpf;


/* CONST */
   half = 0.5;
   one = 1.0;
   fpf = 5.5;
   cof[1] = 76.18009173;
   cof[2] = -86.50532033;
   cof[3] = 24.01409822;
   cof[4] = -1.231739516;
   cof[5] = 0.120858003e-2;
   cof[6] = -0.536382e-5;
   stp = 2.50662827465;
   x = xx-one;
   tmp = x+fpf;
   tmp = (x+half)*log(tmp)-tmp;
   ser = one;
   for (j = 1; j <= 6; j++)
      { x = x+one;
        ser = ser+cof[j]/x;
             }
   return (tmp+log(stp*ser));
}


double betacf(a,b,x)
double a, b, x;
{
double   tem,qap,qam,qab,em,d;
double   bz,bpp,bp,bm,az,app;
double   am,aold,ap;
int      m;
int itmax;
double eps;


   itmax=100;
   eps=3.0e-7;
   am = 1.0;
   bm = 1.0;
   az = 1.0;
   qab = a+b;
```

```
   qap = a+1.0;
   qam = a-1.0;
   bz = 1.0-qab*x/qap;
   for (m = 1; m <= itmax; m++)
    { em = m;
      tem = em+em;
      d = em*(b-m)*x/((qam+tem)*(a+tem));
      ap = az+d*am;
      bp = bz+d*bm;
      d = -(a+em)*(qab+em)*x/((a+tem)*(qap+tem));
      app = ap+d*az;
      bpp = bp+d*bz;
      aold = az;
      am = ap/bpp;
      bm = bp/bpp;
      az = app/bpp;
      bz = 1.0;
      if ((fabs(az-aold)) < (eps*fabs(az))) goto done;
    }
   printf("pause in BETACF\n");
   printf("a or b too big, or itmax too small\n");
done: return az;
}


double betai(a,b,x)
double a, b, x;
{
/*
*****************************************************************
****
This is an incomplete beta function used to generate T and Z
values.  See
Numerical Recipes.
*****************************************************************
****  */

/* VAR */
double bt;

/* BEGIN */
   if (x < 0.0 || x > 1.0)
      printf("pause in routine BETAI");

   if (x == 0.0 || x == 1.0)
     bt = 0.0;
   else
     bt =
exp(gammln(a+b)-gammln(a)-gammln(b)+a*log(x)+b*log(1.0-x));
```

```
   if (x < ((a+1.0)/(a+b+2.0)))
      return (bt*betacf(a,b,x)/a);
   else
      return (1.0-bt*betacf(b,a,1.0-x)/b);
}


double tvalue(tstat,df)
double tstat;
int df;
{

/*
******************************************************************
****
This is original, using a formula developed in Numerical Recipes.
note: this returns the p value of t statistic and df, for one
tailed test.
******************************************************************
*** */
double betai();

return betai(df/2.,1./2.,df/(df+tstat*tstat));
}


double fvalue(fstat,df1,df2)
double fstat;
int df1, df2;
{

/*
******************************************************************
**
This returns the p value for fstat and the two degrees of
freedom.
******************************************************************
**   */
double betai();

return betai(df2/2.,df1/2.,df2/(df2+df1*fstat));
}


/* CALCULATE ORTHOGONAL EXTRA SUMS OF SQUARES */

xby(x, beta, y, xtrassr, m, n)
float x[35][35], beta[35][2], y[35][2], xtrassr[35][2];
int m, n;
{
```

```
float scratch[35][35], scratch1[2][35], scratch3[35][2];
int i;


   transpos(x,scratch,m,n);
   matmulm1(scratch,y,scratch3,n,m,1);
   transpos2(beta,scratch1,n,1);
   matmulnn(scratch3,scratch1,scratch,n,1,n);

   for (i=1; i<=n; i++)
       xtrassr[i][1] = scratch[i][i];

}

/* CALCULATE NON-ORTHOGONAL EXTRA SUMS OF SQUARES */

extrassr(x, beta, y, xtrassr, m, n1, nn, sum)
float x[35][35], beta[35][2], y[35][2], xtrassr[35][2], sum;
int m, n1, nn;
{
float scratch[35][35], scratchy[35][35], scratch1[2][35];
float scratch2[35][35], scratch3[35][2], xpxinv[35][35];
float ss[2][2], betap[2][35], cumssr[35];
int i, j, q, r, n2;


printf("PLEASE BE PATIENT.  IT SOMETIMES TAKES A WHILE TO\n");
printf("CALCULATE INDIVIDUAL SUMS OF SQUARES FOR NON-ORTHOGONAL
DESIGNS\n");

if (n1 > 1)
  {for (i=1; i<=m; i++)
       for (j=1; j<=n1-1; j++)
           scratch2[i][j] = x[i][j];
  }

if (n1 > 1) n2 = n1 - 1;
else n2 = n1;

for (q=n2; q<=nn; q++)
 {
   for (r=1; r<=m; r++)  scratch2[r][q] = x[r][q];

   /* CALCULATE BETAs */

   transpos(scratch2,scratch,m,q);

   matmulmm(scratch,scratch2,scratchy,q,m,q);

   invert(scratchy, xpxinv, q);
```

```
      matmulmm(xpxinv, scratch, scratchy, q, q, m);
      matmulm1(scratchy, y, beta, q, m, 1);
      /* CALCULATE CORRECTED EXTRA SSR */
      transpos2(beta, betap, q, 1);
      transpos(scratch2,scratch,m,q);
      matmul1m(betap, scratch, scratch1, 1, q, m);
      matmul11(scratch1, y, ss, 1, m, 1);
      cumssr[q] = ss[1][1] - (sum*sum)/m;
      if (q >= n1) xtrassr[q][1] = cumssr[q] - cumssr[q-1];
    }
}
```

REGRESS4

```
/* FUNRES12.C SUPPORTS LINEAR REGRESSION WITH LEAST SQUARES */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/* THE FOLLOWING FUNCTIONS INVERT AN NXN MATRIX */

/* FUNCTION INVERT CONTROLS THE INVERSION PROCESS */

invert(float x[35][35], float ainv[35][35], int n)

{
float a[35][35];
float b[35];
int   ipivot[35];
int   i, j, k, iflag, ibeg;

iflag = 1;


for (i=1; i<=n; i++)
     for (j=1; j<=n; j++)
         a[i][j] = x[i][j];


factor(a,n,b,ipivot,iflag);


if (iflag == 2)

   {printf("MATRIX IS SINGULAR\n");
    return;
   }

for (i=1; i<=n; i++)

   { b[i] = 0.;
    }

ibeg = 1;


for (j=1; j<=n; j++)

   { b[j] = 1.;
```

```
      subst(a,ipivot,b,n,ainv,j);
      b[j] = 0.;
      ibeg = ibeg + n;
    }

}


/* FUNCTION FACTOR SUPPORTS THE FUNCTION INVERT  */

/*  factor(a,n,b,ipivot,iflag);  */

factor(float ww[35][35],int n, float d[35],int ipivot[35],
       int iflag)

{

float rowmax, colmax, awikov, temp, ratio;
int i, j, k, istar, nm1, kp1, ip, ipk;

/* INITIALIZE IPIVOT, D  */

for (i=1; i<=n; i++)

   {ipivot[i] = i;
    rowmax = 0.;

    for (j=1; j<=n; j++)

        {
        if (rowmax < fabs(ww[i][j]))
            rowmax = fabs(ww[i][j]);
        }

    if (rowmax == 0.0)
       {iflag = 2;
        return; /*goto done;*/
       }

    d[i] = rowmax;
   }

/* GAUSS ELIMINATION WITH SCALED PARTIAL PIVOTING  */


if (n <= 1) return;

/* DETERMINE PIVOT ROW, THE ROW ISTAR  */
for (k=1; k<=(n-1); k++)
```

```
{ colmax = fabs(ww[k][k]) / d[k];
  istar = k;

  for (i=(k+1); i<=n; i++)

     { awikov = fabs(ww[i][k]) / d[i];

       if (awikov > colmax)
         { colmax = awikov;
           istar = i;
         }
     }

  if (colmax == 0.0)

    { iflag = 2;
      return;
    }

  if (istar > k) /* MAKE K THE PIVOT ROW AND INTERCHANGE */

    { /* iflag = -iflag;  */
      i = ipivot[istar];
      ipivot[istar] = ipivot [k];
      ipivot[k] = i;

      temp = d[istar];
      d[istar] = d[k];
      d[k] = temp;


      for (j=1; j<=n; j++)

         { temp = ww[istar][j];
           ww[istar][j] = ww[k][j];
           ww[k][j] = temp;
         }
    }


/* ELIMINATE X[K] FROM ROW K+1, ..., N  */

      for (i=k+1; i<=n; i++)

         { ww[i][k] = ww[i][k] / ww[k][k];
           ratio = ww[i][k];

           for (j=(k+1); j<=n; j++)

              { ww[i][j] = ww[i][j] - ratio*ww[k][j];
```

```
                }
            }
    }

if (ww[n][n] == 0.0) iflag = 2;


}


/* THE FUNCTION SUBST IS USED AS A SUBSITUTION ALGORTITHM FOR
INVERT */

/*   subst(a,ipivot,b,n,ainv[ibeg]);   */

subst(float ww[35][35],int ipivot[35], float b[35], int n, float
x[35][35],
      int jj)

{

int i, j, k;
int ip, km1, np1, np1mk, kp1;
float sum;




if (n <= 1)
  { x[1][1] = b[1] / ww[1][1];
    return; /* goto done; */
  }

ip = ipivot[1];
x[1][jj] = b[ip];

for (i=2; i<=n; i++)

   { sum = 0.;

     for (j=1; j<=(i-1); j++)
          sum = ww[i][j] * x[j][jj] + sum;

     ip = ipivot[i];
     x[i][jj] = b[ip] - sum;
   }

x[n][jj] = x[n][jj] / ww[n][n];
```

```
for (i=(n-1); i>=1; i--)

   { sum = 0.;

     for (j=(i+1); j<=n; j++)
          sum = ww[i][j] * x[j][jj] + sum;

     x[i][jj] = (x[i][jj]-sum) / ww[i][i];
   }

/* done:*/

}
```

REGOUT.EXE

REGOUT creates the regression aptness and model output. It links five object files, REGOUT1, REGOUT2, REGOUT3, REGOUT4, and REGOUT5.

REGOUT1

```
/* REGOUT.EXE PRODUCES LINEAR REGRESSION OUTPUT */

/* REGOUT1.C CONTROLS LINEAR REGRESSION OUTPUT */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/* MAIN CONTROLS THE OUTPUT PROGRAM REGOUT.EXE */

main ()
{

FILE *infile, *outfile, *file;
int i,j,k,l,m1,n1,n2,nn,p,q,r; /*    COUNTERS      */
int m, n;                      /* # ROWS & COLUMNS    */

int totdf, regdf, errdf;  /* DEGREES OF FREEDOM */
int sstocdf, ssrcdf, ssecdf;

float ssr, ssto, sse;
float ssrc, sstoc, ssec;
float msto, msr, mse;
float msrc, msec;
float freg, fregp, rsq, rsqa;
float sum, ybar, mm;

static float x[35][35];
static float y[35][2];
static float beta[35][2];          /* BETA ESTIMATES   */
float betap[2][35];         /* BETAP             */
float ss[2][2];             /* SS scratch pad */
static float xpxinv[35][35];

static float stats[35][10];    /* REGRESSION STATISTICS   */
static float ssrftest[35];
static float xtrassr[35][2];   /* EXTRA SS */
```

```
static float covcorr[35][35];  /* COVARIANCE/CORRELATION MATRIX
*/

float sspe, mspe;              /* LACK OF FIT PARAMETERS  */
float sslf, mslf;
float flof, flofp;
int level, row, row1;
int facs, totrow;
int reps, creps;
float reps1, creps1;
int c, sspedf, sslfdf;

int gpscreen;
int factors;
int orthog;                /* ORTHOGONAL DESIGN = 1  */
int choice;               /* CHOOSE THE DESIRED TRANSFORMATION */

float wilky, wilk();

char modmenu='1', aptmenu='1', regmenu='1';
char ch;
char outname[24],output[10];


/* READ FROM FILE EXP.DES */

if ((infile  = fopen("regress.out", "r")) == NULL)
  { printf("CAN'T OPEN REGRESSION FILE\n");
    return;
   }

fscanf(infile,"%d\n",&m);
fscanf(infile,"%d\n",&n);
fscanf(infile,"%d\n",&facs);
fscanf(infile,"%d\n",&orthog);
fscanf(infile,"%d\n",&choice);
fscanf(infile,"%d\n",&level);
fscanf(infile,"%d\n",&creps);
fscanf(infile,"%d\n",&reps);
fscanf(infile,"%d\n",&c);
fscanf(infile,"%d\n",&row);

fscanf(infile,"%g\n",&ssrc);
fscanf(infile,"%d\n",&ssrcdf);
fscanf(infile,"%g\n",&msrc);
fscanf(infile,"%g\n",&freg);
fscanf(infile,"%g\n",&ssec);
fscanf(infile,"%d\n",&ssecdf);
fscanf(infile,"%g\n",&msec);
fscanf(infile,"%g\n",&sstoc);
```

```
fscanf(infile,"%d\n",&sstocdf);
fscanf(infile,"%g\n",&fregp);
fscanf(infile,"%g\n",&rsq);
fscanf(infile,"%g\n",&rsqa);

fscanf(infile,"%g\n",&sslf);
fscanf(infile,"%d\n",&sslfdf);
fscanf(infile,"%g\n",&mslf);
fscanf(infile,"%g\n",&sspe);
fscanf(infile,"%d\n",&sspedf);
fscanf(infile,"%g\n",&mspe);
fscanf(infile,"%g\n",&flof);
fscanf(infile,"%g\n",&flofp);

for (i=1; i<=m; i++)
    for (j=1; j<=n; j++)
                 fscanf(infile,"%g %c",&x[i][j],&ch);

fscanf(infile,"\n");

for (i=1; i<=m; i++)
                 fscanf(infile,"%g %c",&y[i][1],&ch);

fscanf(infile,"\n");

for (i=1; i<=m; i++)
    for (j=0; j<=9; j++)
                 fscanf(infile,"%g %c",&stats[i][j],&ch);

fscanf(infile,"\n");

for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
                 fscanf(infile,"%g %c",&xpxinv[i][j],&ch);

fscanf(infile,"\n");

for (i=1; i<=m; i++)
    for (j=0; j<=1; j++)
                 fscanf(infile,"%g %c",&beta[i][j],&ch);

fscanf(infile,"\n");

for (i=1; i<=m; i++)
    for (j=0; j<=2; j++)
                 fscanf(infile,"%g %c",&xtrassr[i][j],&ch);

fscanf(infile,"\n");

for (i=1; i<=m; i++)
```

```
                    fscanf(infile,"%g %c",&ssrftest[i],&ch);
fclose(infile);



strcpy(outname,"data.out");
printf("\nPLEASE PROVIDE A NAME FOR THE OUTPUT FILE:  ");
scanf("%s",outname);
printf("\n");

if ((outfile = fopen(outname, "a")) == NULL)
   {printf("CAN'T OPEN OUTPUT FILE <RETURN>\n");
    return; }

/* CASE STATEMENTS FOR REGRESSION OUTPUT MENU */
while (regmenu != '3')
{

printf("\n\n\n\n\n\n\n\n         \311");
for (k=2; k<=50; k++) printf("\315"); printf("\273\n");
printf("         \272"); for (k=2; k<=50; k++) printf(" ");
printf("\272\n");
printf("         \272               REGRESSION OUTPUT
    \272\n");
printf("         \307"); for (k=2; k<=50; k++) printf("\304");
printf("\266\n");
printf("         \272           (1) APTNESS ASSESSMENT
    \272\n");
printf("         \272           (2) MODEL RESULTS
    \272\n");
printf("         \272           (3) EXIT
    \272\n");
printf("         \272"); for (k=2; k<=50; k++) printf(" ");
printf("\272\n");
printf("         \310"); for (k=2; k<=50; k++) printf("\315");
printf("\274\n\n\n\n\n\n\n\n");

regmenu = getch();

switch(regmenu)
 {
case '1':

/* CASE STATEMENTS FOR APTNESS OUTPUT */
aptmenu = '1';

while (aptmenu != '4')
   {
printf("\n\n\n\n\n\n\n      \311");
```

```
for (k=2; k<=65; k++) printf("\315"); printf("\273\n");
printf("       \272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("       \272                   APTNESS ASSESSMENT
         \272\n");
printf("       \307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("       \272     (1) STANDARDIZED RESIDUALS vs Y-HAT
         \272\n");
printf("       \272     (2) RESIDUALS vs Y-YAT
         \272\n");
printf("       \272     (3) RANKITS vs STANDARDIZED RESIDUALS
(WITH WILK-SHAPIRO)  \272\n");
printf("       \272     (4) EXIT
         \272\n");
printf("       \272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("       \310"); for (k=2; k<=65; k++) printf("\315");
printf("\274\n\n\n\n\n\n\n");

aptmenu = getch();

switch (aptmenu)
     {
case '1':

/* CALCULATE AND PLOT
   Y-HAT (stats[i][6]) vs STANDARDIZED RESIDUALS (stats[i][8])
*/

for (i=1; i<=m; i++)
   { stats[i][7] = y[i][1] - stats[i][6];
     stats[i][8] = stats[i][7]/sqrt(msec);
   }

plot(stats,m,6,8,10.,outfile);

break;

case '2':

/* CALCULATE AND PLOT  Y-HAT (stats[i][6]) vs RESIDUALS
(stats[i][7]) */

for (i=1; i<=m; i++)
    stats[i][7] = y[i][1] - stats[i][6];

plot(stats,m,6,7,10.,outfile);

break;
```

```
case '3':

/* CALCULATE AND PLOT RANKITS (stats[i][9]) vs
   STANDARDIZED RESIDUALS (stats[i][8])  */

for (i=1; i<=m; i++)
   { stats[i][7] = y[i][1] - stats[i][6];
     stats[i][8] = stats[i][7]/sqrt(msec);
   }

/* SORT THE STD RESIDUALS */

for (i=1; i<=m-1; i++)
   { k = i;
     for (nn=7; nn<=8; nn++)
         stats[0][nn] = stats[i][nn];
     for (j = i+1; j<=m; j++)
        { if (stats[j][8] < stats[0][8])
            { k = j;
              for (nn=7; nn<=8; nn++)
                  stats[0][nn] = stats[j][nn];
            }
        }
     for (nn=7; nn<=8; nn++)
        { stats[k][nn] = stats[i][nn];
          stats[i][nn] = stats[0][nn];
        }
   }

/* CALCULATE THE WILK-SHAPIRO STATISTIC  */
wilky = wilk(m, stats);

plot(stats,m,8,9,wilky,outfile);

break;


case '4':

default:

break;
     }
   }

break;
```

```
case '2':

/* CASE STATEMENTS FOR MODEL OUTPUT */
modmenu = '1';

while (modmenu != '8')
  {
printf("\n\n\n\n\n            \311");
for (k=2; k<=50; k++) printf("\315"); printf("\273\n");
printf("            \272"); for (k=2; k<=50; k++) printf(" ");
printf("\272\n");
printf("            \272                  MODEL RESULTS
     \272\n");
printf("            \307"); for (k=2; k<=50; k++) printf("\304");
printf("\266\n");
printf("            \272              (1) ANOVA TABLE
     \272\n");
printf("            \272              (2) COEFFICIENT TABLE
     \272\n");
printf("            \272              (3) VARIANCE-COVARIANCE MATRIX
     \272\n");
printf("            \272              (4) CORRELATION MATRIX
     \272\n");
printf("            \272              (5) LACK OF FIT ANOVA TABLE
     \272\n");
printf("            \272              (6) DESIGN MATRIX
     \272\n");
printf("            \272              (7) INV(X'X) MATRIX
     \272\n");
printf("            \272              (8) EXIT
     \272\n");
printf("            \272"); for (k=2; k<=50; k++) printf(" ");
printf("\272\n");
printf("            \310"); for (k=2; k<=50; k++) printf("\315");
printf("\274\n\n\n\n\n\n");

modmenu = getch();

switch (modmenu)
    {
case '1':

/* ANOVA TABLE FOR CORRECTED SUMS OF SQUARES */
printf("\n\311"); for (k=2; k<=65; k++) printf("\315");
printf("\273\n");
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\272                    ANALYSIS OF VARIANCE
          \272\n");
if (m-n == 0)
```

```
 {printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\272    WARNING ! STATISTICAL TESTS ARE INVALID.  SSE = 0
WITH      \272\n"); }
if (m-n == 0)
printf("\272    df = 0.  THEY HAVE BEEN ARBITRARILY SET TO ONE.
          \272\n");
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272  SOURCE             SS         df            MS
     F    \272\n");
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272  Regression%12.3f      %3d    %12.3f     %8.2f
\272\n",

ssrc,ssrcdf,msrc,freg);
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
if (orthog == 0  && n > 10) n2 = n - 9;
else n2 = 2;
for (j=n; j>=n2; j--)
  { printf("\272    X%-6.0f %12.3f        1    %12.3f     %8.2f
\272\n",
         xtrassr[j][0],xtrassr[j][1],xtrassr[j][1],ssrftest[j]);
    if (j == n-15 || j == n-30)
              { printf("<RETURN>\r"); getch();  printf("
\r"); }
  }
if ( (n >= 11 && n < 15 )||(n2 >= 11 && n2 < 15) )
        { printf("<RETURN>\r"); getch();  printf("         \r");
}

printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272  Error     %12.3f      %3d    %12.3f
\272\n",ssec,ssecdf,msec);
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272  Total     %12.3f      %3d
    \272\n",sstoc,sstocdf);
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("<RETURN>\r"); getch();  printf("         \r");
if (orthog == 1)
 { printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
```

```
   printf("\272  ORTHOGONAL DESIGN
                \272\n");  }
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\272  MODEL F VALUE =  %9.3f    P-VALUE = %7.4f
    \272\n",freg,fregp);
printf("\272  R SQUARED =          %6.3f
         \272\n",rsq);
printf("\272  ADJUSTED R SQUARED =%6.3f
         \272\n",rsqa);
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\310"); for (k=2; k<=65; k++) printf("\315");
printf("\274\n");


/* SEND TO FILE OR PRINTER */

printf("\n1) SEND TO FILE %s     2) SEND TO PRINTER     3)
EXIT",outname);
ch = getch();
printf("\n");

if (ch == '1')
 {
fprintf(outfile,"\n\n\n\311"); for (k=2; k<=65; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\273\n");
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272                      ANALYSIS OF VARIANCE
                    \272\n");
if (m-n == 0)
 {fprintf(outfile,"\272"); for (k=2; k<=65; k++)
fprintf(outfile," "); fprintf(outfile,"\272\n");
fprintf(outfile,"\272    WARNING ! STATISTICAL TESTS ARE INVALID.
SSE = 0 WITH      \272\n"); }
if (m-n == 0)
fprintf(outfile,"\272    df = 0.  THEY HAVE BEEN ARBITRARILY SET
TO ONE.              \272\n");
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
fprintf(outfile,"\272  SOURCE              SS          df
MS             F    \272\n");
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304");  fprintf(outfile,"\266\n");
fprintf(outfile,"\272  Regression%12.3f      %3d    %12.3f
%8.2f  \272\n",
```

```
ssrc,ssrcdf,msrc,freg);
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
if (orthog == 0  && n > 10) n2 = n - 9;
else n2 = 2;
for (j=n; j>=n2; j--)
  { fprintf(outfile,"\272    X%-6.0f %12.3f        1    %12.3f
 %8.2f  \272\n",
         xtrassr[j][0],xtrassr[j][1],xtrassr[j][1],ssrftest[j]);
  }
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
fprintf(outfile,"\272  Error     %12.3f      %3d    %12.3f
      \272\n",ssec,ssecdf,msec);
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
fprintf(outfile,"\272  Total     %12.3f      %3d
            \272\n",sstoc,sstocdf);
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
if (orthog == 1)
 { fprintf(outfile,"\272"); for (k=2; k<=65; k++)
fprintf(outfile," "); fprintf(outfile,"\272\n");
   fprintf(outfile,"\272  ORTHOGONAL DESIGN
                  \272\n");  }
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272  MODEL F VALUE =  %9.3f    P-VALUE = %7.4f
            \272\n",freg,fregp);
fprintf(outfile,"\272  R SQUARED =         %6.3f
                 \272\n",rsq);
fprintf(outfile,"\272  ADJUSTED R SQUARED =%6.3f
                 \272\n",rsqa);
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\310"); for (k=2; k<=65; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\274\n");
 }
else if (ch == '2')
 {
fprintf(stdprn,"\n\n\n\r");
for (k=2; k<=65; k++) fprintf(stdprn,"\315");
fprintf(stdprn,"\n\n\r                    ANALYSIS OF VARIANCE
                   \r\n");
if (m-n == 0)
fprintf(stdprn,"    WARNING ! STATISTICAL TESTS ARE INVALID.  SSE
= 0 WITH \r\n");
if (m-n == 0)
```

```
fprintf(stdprn,"    df = 0.  THEY HAVE BEEN ARBITRARILY SET TO
ONE. \r\n");
fprintf(stdprn,"\r\n");
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n  SOURCE              SS         df
MS           F    \r\n");
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n  Regression%12.3f      %3d    %12.3f
%8.2f  \r\n\r\n",

ssrc,ssrcdf,msrc,freg);
if (orthog == 0  && n > 10) n2 = n - 9;
else n2 = 2;
for (j=n; j>=n2; j--)
  { fprintf(stdprn,"    X%-6.0f %12.3f        1    %12.3f
%8.2f  \r\n",
         xtrassr[j][0],xtrassr[j][1],xtrassr[j][1],ssrftest[j]);
  }
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n  Error     %12.3f      %3d    %12.3f
      \r\n",ssec,ssecdf,msec);
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n  Total     %12.3f      %3d
            \r\n",sstoc,sstocdf);
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n");
if (orthog == 1)
   fprintf(stdprn,"\r\n  ORTHOGONAL DESIGN
             \r\n");
fprintf(stdprn,"\r\n  MODEL F VALUE =  %9.3f    P-VALUE = %7.4f
            \r\n",freg,fregp);
fprintf(stdprn,"  R SQUARED =        %6.3f
             \r\n",rsq);
fprintf(stdprn,"  ADJUSTED R SQUARED =%6.3f
             \r\n",rsqa);
fprintf(stdprn,"\r\n");
for (k=2; k<=65; k++) fprintf(stdprn,"\315");
 }

break;

case '2':

/* VARIANCES (stats[i][2]) and STANDARD ERRORS (stats[i][1])
   OF THE COEFFICIENTS  */

printf("\n\311"); for (k=2; k<=65; k++) printf("\315");
printf("\273\n");
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
```

```
printf("\272                            COEFFICIENT TABLE
          \272\n");
if (m-n == 0)
 {printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\272    WARNING ! STATISTICAL TESTS ARE INVALID.  SSE = 0
WITH      \272\n"); }
if (m-n == 0)
printf("\272    df = 0.  THEY HAVE BEEN ARBITRARILY SET TO ONE.
          \272\n");
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272 VARIABLE     VALUE   STD ERROR   VARIANCE  STUDENT-T
P-VALUE  \272\n");
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");

k=1;
for (i=1; i<=n; i++)
   { printf("\272 X%-6.0f%11.3f  %10.3f %10.3f   %7.3f   %7.4f
\272\n",

stats[i][5],stats[i][0],stats[i][1],stats[i][2],stats[i][3],stats
[i][4]);
     k = k+1;
     if (k == 15 || k == 32)
                 {    printf("<RETURN>\r"); getch();  printf("
     \r"); }
   }
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\310"); for (k=2; k<=65; k++) printf("\315");
printf("\274\n");

/* SEND TO FILE OR PRINTER */

printf("\n1) SEND TO FILE %s     2) SEND TO PRINTER     3)
EXIT",outname);
ch = getch();
printf("\n");

if (ch == '1')
 {
fprintf(outfile,"\n\n\n\311"); for (k=2; k<=65; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\273\n");
```

```
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272                           COEFFICIENT TABLE
                   \272\n");
if (m-n == 0)
 {fprintf(outfile,"\272"); for (k=2; k<=65; k++)
fprintf(outfile," "); fprintf(outfile,"\272\n");
fprintf(outfile,"\272    WARNING ! STATISTICAL TESTS ARE INVALID.
SSE = 0 WITH      \272\n"); }
if (m-n == 0)
fprintf(outfile,"\272    df = 0.  THEY HAVE BEEN ARBITRARILY SET
TO ONE.             \272\n");
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
fprintf(outfile,"\272 VARIABLE     VALUE   STD ERROR   VARIANCE
STUDENT-T  P-VALUE  \272\n");
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");

k=1;
for (i=1; i<=n; i++)
   { fprintf(outfile,"\272 X%-6.0f%11.3f  %10.3f %10.3f   %7.3f
%7.4f  \272\n",

stats[i][5],stats[i][0],stats[i][1],stats[i][2],stats[i][3],stats
[i][4]);
     k = k+1;
   }
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\310"); for (k=2; k<=65; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\274\n");

 }
else if (ch == '2')
 {

fprintf(stdprn,"\n\n\n\r");
for (k=2; k<=65; k++) fprintf(stdprn,"\315");
fprintf(stdprn,"\r\n\n                          COEFFICIENT TABLE
                    \n\r");
if (m-n == 0)
fprintf(stdprn,"\n    WARNING ! STATISTICAL TESTS ARE INVALID.
SSE = 0 WITH      \n\r");
if (m-n == 0)
```

```
fprintf(stdprn,"    df = 0.  THEY HAVE BEEN ARBITRARILY SET TO
ONE.            \n\n\r");
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\n\r VARIABLE     VALUE   STD ERROR   VARIANCE
STUDENT-T  P-VALUE  \n\r");
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\n\n\r");

k=1;
for (i=1; i<=n; i++)
   { fprintf(stdprn," X%-6.0f%11.3f  %10.3f %10.3f   %7.3f
%7.4f  \n\r",

stats[i][5],stats[i][0],stats[i][1],stats[i][2],stats[i][3],stats
[i][4]);
     k = k+1;
   }
fprintf(stdprn,"\n\r");
for (k=2; k<=65; k++) fprintf(stdprn,"\315");

 }

break;


case '3':


/* CALCULATE VARIANCE-COVARIANCE MATRIX */
covmat(msec,xpxinv,covcorr,n);

covar(beta,covcorr,n,outname,outfile);

break;


case '4':


corrmat(msec,xpxinv,orthog,covcorr,n);

corr(beta,covcorr,n,outname,outfile);

break;


case '5':

if (creps > 0 || reps > 0)
 {
```

```
/* ANALYSIS OF VARIANCE WITH LACK OF FIT */

printf("\n\311"); for (k=2; k<=65; k++) printf("\315");
printf("\273\n");
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\272           ANALYSIS OF VARIANCE WITH LACK OF FIT
          \272\n");
if (m-n == 0)
 {printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\272    WARNING ! STATISTICAL TESTS ARE INVALID.  SSE = 0
WITH      \272\n"); }
if (m-n == 0)
printf("\272    df = 0.  THEY HAVE BEEN ARBITRARILY SET TO ONE.
          \272\n");
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272  SOURCE            SS         df            MS
     F    \272\n");
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272  Regression%12.3f      %3d    %12.3f     %8.2f
\272\n",

ssrc,ssrcdf,msrc,freg);
printf("\272  Error     %12.3f      %3d    %12.3f
\272\n",ssec,ssecdf,msec);
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272    Lack of Fit%9.3f      %3d    %12.3f     %8.2f
\272\n",

sslf,sslfdf,mslf,flof);
printf("\272    Pure Error %9.3f      %3d    %12.3f
\272\n",

sspe,sspedf,mspe);
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
printf("\272  Total     %12.3f      %3d
    \272\n",sstoc,sstocdf);
printf("\307"); for (k=2; k<=65; k++) printf("\304");
printf("\266\n");
if (orthog == 1)
 { printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
```

```
   printf("\272  ORTHOGONAL DESIGN
              \272\n"); }
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\272  MODEL F VALUE =        %9.3f    P-VALUE = %7.4f
    \272\n",freg,fregp);
printf("\272  LACK OF FIT F VALUE =  %9.3f    P-VALUE = %7.4f
    \272\n",flof,flofp);
printf("\272  R SQUARED =         %6.3f
         \272\n",rsq);
printf("\272  ADJUSTED R SQUARED =%6.3f
         \272\n",rsqa);
printf("\272"); for (k=2; k<=65; k++) printf(" ");
printf("\272\n");
printf("\310"); for (k=2; k<=65; k++) printf("\315");
printf("\274\n");


/* SEND TO FILE OR PRINTER */

printf("\n1) SEND TO FILE %s     2) SEND TO PRINTER     3)
EXIT",outname);
ch = getch();
printf("\n");

if (ch == '1')
 {
fprintf(outfile,"\n\n\n\311"); for (k=2; k<=65; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\273\n");
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272          ANALYSIS OF VARIANCE WITH LACK OF
FIT                \272\n");
if (m-n == 0)
 {fprintf(outfile,"\272"); for (k=2; k<=65; k++)
fprintf(outfile," "); fprintf(outfile,"\272\n");
fprintf(outfile,"\272    WARNING ! STATISTICAL TESTS ARE INVALID.
SSE = 0 WITH       \272\n"); }
if (m-n == 0)
fprintf(outfile,"\272    df = 0.  THEY HAVE BEEN ARBITRARILY SET
TO ONE.            \272\n");
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
fprintf(outfile,"\272  SOURCE             SS         df
MS             F    \272\n");
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304");  fprintf(outfile,"\266\n");
```

```
fprintf(outfile,"\272  Regression%12.3f      %3d    %12.3f
%8.2f  \272\n",

ssrc,ssrcdf,msrc,freg);
fprintf(outfile,"\272  Error     %12.3f      %3d    %12.3f
       \272\n",

ssec,ssecdf,msec);
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304");  fprintf(outfile,"\266\n");
fprintf(outfile,"\272    Lack of Fit%9.3f      %3d    %12.3f
%8.2f  \272\n",

sslf,sslfdf,mslf,flof);
fprintf(outfile,"\272    Pure Error %9.3f      %3d    %12.3f
         \272\n",

sspe,sspedf,mspe);
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304");  fprintf(outfile,"\266\n");
fprintf(outfile,"\272  Total     %12.3f      %3d
             \272\n",sstoc,sstocdf);
fprintf(outfile,"\307"); for (k=2; k<=65; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
if (orthog == 1)
 { fprintf(outfile,"\272"); for (k=2; k<=65; k++)
fprintf(outfile," "); fprintf(outfile,"\272\n");
   fprintf(outfile,"\272  ORTHOGONAL DESIGN
                      \272\n");  }
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272  MODEL F VALUE =         %9.3f    P-VALUE =
%7.4f        \272\n",freg,fregp);
fprintf(outfile,"\272  LACK OF FIT F VALUE =  %9.3f    P-VALUE =
%7.4f        \272\n",flof,flofp);
fprintf(outfile,"\272  R SQUARED =         %6.3f
                 \272\n",rsq);
fprintf(outfile,"\272  ADJUSTED R SQUARED =%6.3f
                 \272\n",rsqa);
fprintf(outfile,"\272"); for (k=2; k<=65; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\310"); for (k=2; k<=65; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\274\n");

 }

else if (ch == '2')

 {
fprintf(stdprn,"\n\n\n\r");
```

```
for (k=2; k<=65; k++) fprintf(stdprn,"\315");
fprintf(stdprn,"\n\n\r     ANALYSIS OF VARIANCE WITH LACK OF FIT
    \r\n");
if (m-n == 0)
fprintf(stdprn,"    WARNING ! STATISTICAL TESTS ARE INVALID.  SSE
= 0 WITH \r\n");
if (m-n == 0)
fprintf(stdprn,"    df = 0.  THEY HAVE BEEN ARBITRARILY SET TO
ONE. \r\n");
fprintf(stdprn,"\r\n");
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n  SOURCE            SS          df
MS            F     \r\n");
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n  Regression%12.3f      %3d    %12.3f
%8.2f  ",

ssrc,ssrcdf,msrc,freg);
fprintf(stdprn,"\r\n  Error     %12.3f      %3d    %12.3f
      \r\n",ssec,ssecdf,msec);
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n    Lack of Fit%9.3f      %3d    %12.3f
%8.2f ",sslf,sslfdf,mslf,flof);
fprintf(stdprn,"\r\n    Pure Error %9.3f      %3d
%12.3f\r\n",sspe,sspedf,mspe);
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n  Total     %12.3f      %3d
\r\n",sstoc,sstocdf);
for (k=2; k<=65; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\r\n");
if (orthog == 1)
   fprintf(stdprn,"\r\n  ORTHOGONAL DESIGN
  \r\n");
fprintf(stdprn,"\r\n  MODEL F VALUE =      %9.3f    P-VALUE =
%7.4f     \r\n",freg,fregp);
fprintf(stdprn,"  LACK OF FIT F VALUE =%9.3f    P-VALUE = %7.4f
  \r\n\n",flof,flofp);
fprintf(stdprn,"  R SQUARED =        %6.3f
  \r\n",rsq);
fprintf(stdprn,"  ADJUSTED R SQUARED =%6.3f
  \r\n",rsqa);
fprintf(stdprn,"\r\n");
for (k=2; k<=65; k++) fprintf(stdprn,"\315");

 }

}
else
 {printf("LACK OF FIT UNAVAILABLE WITHOUT REPS <RETURN>\n");
  getch();
```

```
 }

break;


case '6':

/* PRINT DESIGN MATRIX, Y, AND Y-HAT  */

printf("\n\311"); for (k=2; k<=60; k++) printf("\315");
printf("\273\n");
printf("\272"); for (k=2; k<=60; k++) printf(" ");
printf("\272\n");
printf("\272                      DESIGN MATRIX
     \272\n");
if (level == 3)
{ printf("\272      NOTE: QUADRATICS TERMS HAVE BEEN CORRECTED
       \272\n");
printf("\272        SO THEY ARE ORTHOGONAL TO THE MEAN
     \272\n"); }
printf("\272"); for (k=2; k<=60; k++) printf(" ");
printf("\272\n");

n1 = 1;
if (n+2 > 8) n2 = 8;
   else n2 = n+2;
while (n1 <= n+2)
   {

printf("\307"); for (k=2; k<=60; k++) printf("\304");
printf("\266\n");
for (k=1; k<=60; k++) printf(" "); printf("\272\r");
printf("\272       ");
   for (i=n1; i<=n2; i++)
       {  if (i <= n)   printf("X%-5.0f",beta[i][0]);
          if (i == n+1) printf("     Y");
          if (i == n+2) printf("       Y-HAT");
       }
printf("\n\307"); for (k=2; k<=60; k++) printf("\304");
printf("\266\n");

   for (i=1; i<=m; i++)
     { for (k=1; k<=60; k++) printf(" "); printf("\272\r");
       printf("\272  %3d",i);
       for (j=n1; j<=n2; j++)
          { if (j <= n)   printf("%6.2f",x[i][j]);
            if (j == n+1) printf("%10.2f",y[i][1]);
            if (j == n+2) printf("%10.2f",stats[i][6]);
          }
```

```
        printf("\n");
        if (i == 17 || i == 34)
           { printf("<RETURN>\r"); getch();  printf("
\r"); }
      }
   printf("\272"); for (k=2; k<=60; k++) printf(" ");
printf("\272\n");
   printf("\310"); for (k=2; k<=60; k++) printf("\315");
printf("\274\n\n");
   printf("<RETURN>\r"); getch();  printf("         \r");
   n1 = n1+8;
    n2 = n2+8;
    if (n2 > n+2) n2 = n+2;
    }

/* SEND TO FILE OR PRINTER */
printf("1) SEND TO FILE %s     2) SEND TO PRINTER     3)
EXIT",outname);
ch = getch();
printf("\n");

if (ch == '1')
 {

fprintf(outfile,"\n\n\n\311"); for (k=2; k<=60; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\273\n");
fprintf(outfile,"\272"); for (k=2; k<=60; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272                    DESIGN MATRIX
             \272\n");
if (level == 3)
{ fprintf(outfile,"\272      NOTE: QUADRATICS TERMS HAVE BEEN
CORRECTED          \272\n");
fprintf(outfile,"\272          SO THEY ARE ORTHOGONAL TO THE MEAN
             \272\n"); }
fprintf(outfile,"\272"); for (k=2; k<=60; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");

n1 = 1;
if (n+2 > 8) n2 = 8;
   else n2 = n+2;
while (n1 <= n+2)
   {

fprintf(outfile,"\307"); for (k=2; k<=60; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
for (k=1; k<=60; k++) fprintf(outfile," ");
fprintf(outfile,"\272\r");
fprintf(outfile,"\272       ");
   for (i=n1; i<=n2; i++)
```

```
        {  if (i <= n)   fprintf(outfile,"X%-5.0f",beta[i][0]);
           if (i == n+1) fprintf(outfile,"     Y");
           if (i == n+2) fprintf(outfile,"       Y-HAT");
        }
   fprintf(outfile,"\n\307"); for (k=2; k<=60; k++)
   fprintf(outfile,"\304"); fprintf(outfile,"\266\n");

      for (i=1; i<=m; i++)
        { for (k=1; k<=60; k++) fprintf(outfile," ");
   fprintf(outfile,"\272\r");
          fprintf(outfile,"\272  %3d",i);
          for (j=n1; j<=n2; j++)
             { if (j <= n)   fprintf(outfile,"%6.2f",x[i][j]);
               if (j == n+1) fprintf(outfile,"%10.2f",y[i][1]);
               if (j == n+2) fprintf(outfile,"%10.2f",stats[i][6]);
             }
          fprintf(outfile,"\n");
        }
      fprintf(outfile,"\272"); for (k=2; k<=60; k++)
   fprintf(outfile," "); fprintf(outfile,"\272\n");
      fprintf(outfile,"\310"); for (k=2; k<=60; k++)
   fprintf(outfile,"\315"); fprintf(outfile,"\274\n\n\n");
      n1 = n1+8;
      n2 = n2+8;
      if (n2 > n+2) n2 = n+2;
      }

    }
   else if (ch == '2')
    {
   fprintf(stdprn,"\n\n\n\r");
   for (k=1; k<=60; k++) fprintf(stdprn,"\315");
   fprintf(stdprn,"\r\n\n                     DESIGN MATRIX  \n\r");
   if (level == 3)
   { fprintf(stdprn,"     NOTE: QUADRATICS TERMS HAVE BEEN
   CORRECTED  \r\n");
   fprintf(stdprn,"       SO THEY ARE ORTHOGONAL TO THE MEAN
    \r\n"); }
   fprintf(stdprn,"\n");

   n1 = 1;
   if (n+2 > 8) n2 = 8;
      else n2 = n+2;
   while (n1 <= n+2)
      {

   for (k=1; k<=60; k++) fprintf(stdprn,"\304");
   fprintf(stdprn,"\n\r        ");
      for (i=n1; i<=n2; i++)
        {  if (i <= n)   fprintf(stdprn,"X%-5.0f",beta[i][0]);
```

```
            if (i == n+1) fprintf(stdprn,"      Y");
            if (i == n+2) fprintf(stdprn,"       Y-HAT");
         }
fprintf(stdprn,"\n\r");
for (k=1; k<=60; k++) fprintf(stdprn,"\304");

   for (i=1; i<=m; i++)
     { fprintf(stdprn,"\n\r  %3d",i);
       for (j=n1; j<=n2; j++)
          { if (j <= n)   fprintf(stdprn,"%6.2f",x[i][j]);
            if (j == n+1) fprintf(stdprn,"%10.2f",y[i][1]);
            if (j == n+2) fprintf(stdprn,"%10.2f",stats[i][6]);
          }
     }
   fprintf(stdprn,"\n\r");
   for (k=1; k<=60; k++) fprintf(stdprn,"\315");
   fprintf(stdprn,"\r\n\n");
   n1 = n1+8;
   n2 = n2+8;
   if (n2 > n+2) n2 = n+2;
   }
 }

break;


case '7':

/* PRINT THE INV(X'X) MATRIX */

xpxinvout(xpxinv, beta, n, outname, outfile);

break;


case '8':

default:

break;
    }
  }

break;


case '3':
```

```
break;

default:

break;
 }
}

fclose(outfile);

}
```

REGOUT2

```
/* REGOUT2.C SUPPORTS LINEAR REGRESSION OUTPUT */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/*  FUNCTION MATMULmm MULTIPLIES MATRIX a (mxn) BY MATRIX b (nxm)
    AND CREATES MATRIX c (mxm) */

matmulmm(float a[35][35], float b[35][35], float c[35][35],
       int m, int n, int p)


{
int i, j, k;
float sum;


   for (i=1; i<=m; i++)

      {for (j=1; j<=p; j++)

          {sum = 0.0;

           for (k=1; k<=n; k++)
              { sum = sum + a[i][k] * b[k][j]; }

           c[i][j] = sum;
          }
      }

}


/* FUNCTION WILK CALCULATES THE WILK SHAPIRO STATISTIC FOR
   NORMALITY  */

float wilk(n,res)
int n;
float res[35][10];
{

float b[35], resbar, w, mpm, rtmpm, sum, wilky;
int i;

resbar = 0;
```

```
sum = 0;
mpm = 0;
w = 0;

for (i=1; i<=n; i++)
     resbar = resbar + res[i][8];

resbar = resbar/n;

for (i=1; i<=n; i++)
     sum = sum + (res[i][8] - resbar)*(res[i][8] - resbar);

if (sum < .0001) sum = .001;

for (i=1; i<=n; i++)
     mpm = mpm + res[i][9]*res[i][9];

rtmpm = sqrt(mpm);

for (i=1; i<=n; i++)
    b[i] = res[i][9]/rtmpm;

for (i=1; i<=n; i++)
    w = w + b[i]*res[i][8];

w = w*w;

wilky = w/sum;

return wilky;

}


/* SCATTERPLOT PROGRAM */

plot(stats,m,xcol,ycol,wilk,outfile)
float stats[35][10], wilk;
int m, xcol, ycol;
FILE *outfile;
{
char label[10][20], save;
/* char far *ptr = (char far *) 0xB8000000;  pointer to CGA
memory */
int plot[35][10];
float x, y, min, max;
int minx, maxx, rangex, miny, maxy, rangey;
int i, j;
```

```
strcpy(label[1],"STD ERROR");
strcpy(label[2],"VARIANCE");
strcpy(label[3],"T STATISTIC");
strcpy(label[4],"P-VALUE");
strcpy(label[5],"NAME");
strcpy(label[6],"Y-HAT");
strcpy(label[7],"RESID");
strcpy(label[8],"STD RES");
strcpy(label[9],"RANKITS");

/* find minimum, maximum, and range of the Xs */

min = stats[1][xcol];
for (i=2; i<=m; i++)
    if (stats[i][xcol] < min)   min = stats[i][xcol];
minx = floor(min);

max = stats[1][xcol];
for (i=2; i<=m; i++)
    if (stats[i][xcol] > max)   max = stats[i][xcol];
maxx = ceil(max);

rangex = maxx - minx;

if (rangex <= 1) rangex = 1;
else if (rangex > 10 && rangex <= 20) rangex = 20;
else if (rangex > 20 && rangex <= 50) rangex = 50;
else if (rangex > 50 && rangex <= 100) rangex = 100;

/* transform Xs into plotting coordinates */

if (minx == 0)
   {
    for (i=1; i<=m; i++)
       plot[i][xcol] = floor(stats[i][xcol]*200./rangex + 51);
   }
else
   {
    for (i=1; i<=m; i++)
       plot[i][xcol] = floor( (stats[i][xcol] - minx)*200./rangex
+ 50);
   }

/* find minimum, maximum, and range of the Ys */

min = stats[1][ycol];
for (i=2; i<=m; i++)
    if (stats[i][ycol] < min)   min = stats[i][ycol];
miny = floor(min);
```

```
max = stats[1][ycol];
for (i=2; i<=m; i++)
    if (stats[i][ycol] > max)   max = stats[i][ycol];
maxy = ceil(max);

rangey = maxy - miny;

if (rangey <= 1) rangey = 1;
else if (rangey > 8 && rangey <= 10) rangey = 10;
else if (rangey > 10 && rangey <= 20) rangey = 20;
else if (rangey > 20 && rangey <= 50) rangey = 50;
else if (rangey > 50 && rangey <= 100) rangey = 100;

/* transform Ys into plotting coordinates */

if (miny == 0)
   {
    for (i=1; i<=m; i++)
       plot[i][ycol] = floor(169. - stats[i][ycol]*150./rangey);
   }
else
   {
    for (i=1; i<=m; i++)
       plot[i][ycol] = floor(169. - (stats[i][ycol] -
miny)*150./rangey);
   }


mode(4);
palette(1);


gridx(minx, rangex,label,xcol);
gridy(miny, rangey, wilk,label,xcol,ycol);

for (i=1; i<=m; i++)
   {mempoint(plot[i][ycol]-1, plot[i][xcol],2);
    mempoint(plot[i][ycol]+1, plot[i][xcol],2);
    mempoint(plot[i][ycol], plot[i][xcol],2);
    mempoint(plot[i][ycol], plot[i][xcol]-1,2);
    mempoint(plot[i][ycol], plot[i][xcol]+1,2);
   }

save = getch();

/* PLACE CODE HERE TO SEND PLOT TO FILE OR PRINTER */

mode(3);
```

```
}

gridx(minx,rangex,label,xcol)
int minx, rangex, xcol;
char label[][20];

{
 register int t, i, j;
 float x1, x2, x3, x4, x5, x6;

 x1 = minx;
 x2 = minx + .2*rangex;
 x3 = minx + .4*rangex;
 x4 = minx + .6*rangex;
 x5 = minx + .8*rangex;
 x6 = minx + rangex;


 gotoxy(22,5); printf("%.1f", x1);
 gotoxy(22,10); printf("%.1f", x2);
 gotoxy(22,15); printf("%.1f", x3);
 gotoxy(22,20); printf("%.1f", x4);
 gotoxy(22,25); printf("%.1f", x5);
 gotoxy(22,30); printf("%.1f",x6);
 gotoxy(20,32); printf(label[xcol]);

 line(170, 50, 170, 250, 1);

 j = 50;
 while (j<=250)
    { for (i=170; i<=173; i++)
          mempoint(i,j,1);
      j = j + 40;
    }

}

gridy(miny,rangey,wilk,label,xcol,ycol)
int miny, rangey, xcol, ycol;
float wilk;
char label[][20];
{
 register int t,i,j;
 float y1, y2, y3, y4, y5, y6;

 y1 = miny;
 y2 = miny + .2*rangey;
 y3 = miny + .4*rangey;
 y4 = miny + .6*rangey;
 y5 = miny + .8*rangey;
```

```
 y6 = miny + rangey;


 gotoxy( 1,5); printf(label[ycol]);
 gotoxy( 2,17); printf(label[xcol]);
                 printf(" vs "); printf(label[ycol]);
 gotoxy( 2,0); printf("%.1f", y6);
 gotoxy( 6,0); printf("%.1f", y5);
 gotoxy(10,0); printf("%.1f", y4);
 gotoxy(14,0); printf("%.1f", y3);
 gotoxy(17,0); printf("%.1f", y2);
 gotoxy(21,0); printf("%.1f", y1);
 if (wilk < 2.0) { gotoxy(24,0); printf("WILK SHÀPIRO =
%.3f",wilk); }
/* gotoxy(24,22); printf("1)TO FILE  2)EXIT"); */

 line(20, 50, 170, 50, 1);

 i = 20;
 while (i<=170)
    { for (j=48; j<=50; j++)
          mempoint(i,j,1);
      i = i + 30;
    }

}

line(startx, starty, endx, endy, color)
int startx, starty, endx, endy, color;
{
register int t, distance;
int x=0, y=0, deltax, deltay;
int incx, incy;

/* compute the distances in both directions */
deltax = endx - startx;
deltay = endy - starty;

/* compute direction (0 means vert or hor)  */
if (deltax > 0) incx = 1;
else if (deltax == 0) incx = 0;
else incx = -1;

if (deltay > 0) incy = 1;
else if (deltay == 0) incy = 0;
else incy = -1;

/* determine which distance is greater */
deltax = abs(deltax);
deltay = abs(deltay);
```

```
if (deltax > deltay) distance = deltax;
else distance = deltay;

/* draw the line */
for (t=0; t<=distance+1; t++)
 { mempoint(startx, starty, color);
   x+= deltax;
   y+= deltay;
   if (x > distance)
     { x-=distance;
       startx+=incx;
     }
   if (y > distance)
     { y-=distance;
       starty+=incy;
     }
 }
}

/* write point to screen */
mempoint(x, y, colorcode)
int x, y, colorcode;
{
union mask {
   char c[2];
   int i;  } bitmask;
int i, index, bitposition;
unsigned char t;
char xor;
char far *ptr = (char far *) 0xB8000000;

bitmask.i = 0xFF3F;

/* check range for mode 4 */
if (x < 0 || x > 199 || y < 0 || y > 319) return;

xor = colorcode & 128;
colorcode = colorcode & 127;

/* set bitmask and colocode bits to right location */
bitposition = y%4;
colorcode <<= 2*(3-bitposition);
bitmask.i >>= 2*bitposition;

/* find the correct byte in screen memory */
index = x*40 +(y >> 2);
if (x % 2) index +=8152;

/* write the color */
if (!xor)
```

```
  { t = *(ptr+index) & bitmask.c[0];
    *(ptr+index) = t | colorcode;
  }
else
  { t = *(ptr+index) | (char)0;
    *(ptr+index) = t ^ colorcode;
  }
}

/* set video mode */
mode(modecode)
int modecode;
{
union REGS r;
r.h.al = modecode;
r.h.ah = 0;
int86(0x10, &r, &r);
}

/* send cursor to x,y */
gotoxy(x, y)
int x, y;
{

union REGS r;
r.h.ah = 2;
r.h.dl = y;
r.h.dh = x;
r.h.bh = 0;
int86(0x10, &r, &r);
}

/* set the palette */
palette(pnum)
int pnum;
{

union REGS r;
r.h.bh = 1;
r.h.bl = pnum;
r.h.ah = 11;
int86(0x10, &r, &r);
}
```

REGOUT3

```
/* REGOUT3.C SUPPORTS LINEAR REGRESSION OUTPUT */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/* FUNTION CORR CALCULATES THE CORRELATION MATRIX */
corr(beta,scratchy,n,outname,outfile)
float beta[35][2], scratchy[35][35];
int n;
char outname[24];
FILE *outfile;
{

int i, j, k, n1, n2;
char ch;


printf("\n\311"); for (k=2; k<=67; k++) printf("\315");
printf("\273\n");
printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");
printf("\272                        CORRELATION MATRIX
            \272\n");
printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");

n1 = 1;
if (n > 8) n2 = 8;
   else n2 = n;
while (n1 <= n)
   {

printf("\307"); for (k=2; k<=67; k++) printf("\304");
printf("\266\n");
for (k=1; k<=67; k++) printf(" "); printf("\272\r");
printf("\272         ");
for (i=n1; i<=n2; i++)  printf(" X%-5.0f",beta[i][0]);
printf("\n\307"); for (k=2; k<=67; k++) printf("\304");
printf("\266\n");

   for (i=1; i<=n; i++)
     { for (k=1; k<=67; k++) printf(" "); printf("\272\r");
       printf("\272  X%-5.0f",beta[i][0]);
       for (j=n1; j<=n2; j++)
           {printf("%7.3f",scratchy[i][j]);}
```

```
        printf("\n");
        if (i == 17 || i == 34)
           { printf("<RETURN>\r"); getch(); printf("
\r"); }
      }
   printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");
   printf("\310"); for (k=2; k<=67; k++) printf("\315");
printf("\274\n\n");
   printf("<RETURN>\r"); getch(); printf("          \r");
   n1 = n1+8;
    n2 = n2+8;
    if (n2 > n) n2 = n;
    }

/* SEND TO FILE OR PRINTER */
printf("1) SEND TO FILE %s     2) SEND TO PRINTER     3)
EXIT",outname);
ch = getch();
printf("\n");

if (ch == '1')
 {

fprintf(outfile,"\n\n\n\311"); for (k=2; k<=67; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\273\n");
fprintf(outfile,"\272"); for (k=2; k<=67; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272                              CORRELATION MATRIX
                     \272\n");
fprintf(outfile,"\272"); for (k=2; k<=67; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");

n1 = 1;
if (n > 8) n2 = 8;
   else n2 = n;
while (n1 <= n)
    {

fprintf(outfile,"\307"); for (k=2; k<=67; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
for (k=1; k<=67; k++) fprintf(outfile," ");
fprintf(outfile,"\272\r");
fprintf(outfile,"\272        ");
for (i=n1; i<=n2; i++)  fprintf(outfile," X%-5.0f",beta[i][0]);
fprintf(outfile,"\n\307"); for (k=2; k<=67; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");

   for (i=1; i<=n; i++)
```

```
      { for (k=1; k<=67; k++) fprintf(outfile," ");
fprintf(outfile,"\272\r");
        fprintf(outfile,"\272  X%-5.0f",beta[i][0]);
        for (j=n1; j<=n2; j++)
            {fprintf(outfile,"%7.3f",scratchy[i][j]);}
        fprintf(outfile,"\n");
      }
   fprintf(outfile,"\272"); for (k=2; k<=67; k++)
fprintf(outfile," "); fprintf(outfile,"\272\n");
   fprintf(outfile,"\310"); for (k=2; k<=67; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\274\n\n\n");
   n1 = n1+8;
    n2 = n2+8;
    if (n2 > n) n2 = n;
    }

  }
else if (ch == '2')
  {

   corrprt(scratchy, beta, n);

  }


}

/* SEND CORRLELATION MATRIX TO PRINTER */

corrprt(scratchy, beta, n)
float scratchy[35][35], beta[35][2];
int n;
{
int i, j, k, n1, n2;
char ch;

fprintf(stdprn,"\n\n\n\r");
for (k=2; k<=67; k++) fprintf(stdprn,"\315");
fprintf(stdprn,"\r\n\n                        CORRELATION MATRIX
    \n\r");

n1 = 1;
if (n > 8) n2 = 8;
   else n2 = n;
while (n1 <= n)
    {

fprintf(stdprn,"\r\n");
for (k=2; k<=67; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\n\r         ");
```

```
   for (i=n1; i<=n2; i++)
           fprintf(stdprn," X%-5.0f",beta[i][0]);
fprintf(stdprn,"\r\n");
for (k=2; k<=67; k++) fprintf(stdprn,"\304");

   for (i=1; i<=n; i++)
     { fprintf(stdprn,"\r\n  X%-5.0f",beta[i][0]);
       for (j=n1; j<=n2; j++)
           {fprintf(stdprn,"%7.3f",scratchy[i][j]);}
     }
   fprintf(stdprn,"\r\n\n");
   for (k=2; k<=67; k++) fprintf(stdprn,"\315");
   fprintf(stdprn,"\r\n\n\n");
   n1 = n1+8;
    n2 = n2+8;
    if (n2 > n) n2 = n;
   }
}


/* FUNTION CORRMAT COMPUTES THE CORRELATION MATRIX */

corrmat(msec,xpxinv,orthog,scratchy,n)
float msec, xpxinv[35][35],scratchy[35][35];
int orthog,n;
{

float scratch[35][35];
float scratch2[35][35], scratch4[35][35];
int i, j, k, n1, n2;
char ch;

for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
         scratch[i][j] = msec * xpxinv[i][j];

if (orthog == 1)
  {
    for (i=1; i<=n; i++)
        for (j=1; j<=n; j++)
            scratchy[i][j] = 0;

    for (i=1; i<=n; i++)
        scratchy[i][i] = 1.;
  }

else

  {
```

```
    for (i=1; i<=n; i++)
        for (j=1; j<=n; j++)
                  scratch4[i][j] = 0;

    for (i=1; i<=n; i++)
         scratch4[i][i] = 1./sqrt(scratch[i][i]);

    matmulmm(scratch4,scratch,scratch2,n,n,n);

    matmulmm(scratch2,scratch4,scratchy,n,n,n);

   }
}
```

REGOUT4

```
/* REGOUT4.C SUPPORTS LINEAR REGRESSION OUTPUT */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/* FUNTION COVAR CALCULATES THE VAR-COVARIANCE MATRIX */
covar(beta,scratch,n,outname,outfile)
float beta[35][2], scratch[35][35];
int n;
char outname[24];
FILE *outfile;
{

int i, j, k, n1, n2;
char ch;


printf("\n\311"); for (k=2; k<=67; k++) printf("\315");
printf("\273\n");
printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");
printf("\272                     VARIANCE-COVARIANCE MATRIX
            \272\n");
printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");

n1 = 1;
if (n > 7) n2 = 7;
   else n2 = n;
while (n1 <= n)
   {

printf("\307"); for (k=2; k<=67; k++) printf("\304");
printf("\266\n");
for (k=1; k<=67; k++) printf(" "); printf("\272\r");
printf("\272         ");
for (i=n1; i<=n2; i++)  printf(" X%-6.0f",beta[i][0]);
printf("\n\307"); for (k=2; k<=67; k++) printf("\304");
printf("\266\n");

   for (i=1; i<=n; i++)
     { for (k=1; k<=67; k++) printf(" "); printf("\272\r");
       printf("\272  X%-5.0f",beta[i][0]);
       for (j=n1; j<=n2; j++)
           {printf("%8.3f",scratch[i][j]);}
```

```
        printf("\n");
        if (i == 17 || i == 34)
           { printf("<RETURN>\r"); getch(); printf("
\r"); }
     }
   printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");
   printf("\310"); for (k=2; k<=67; k++) printf("\315");
printf("\274\n\n");
   printf("<RETURN>\r");  getch();  printf("          \r");
   n1 = n1+7;
    n2 = n2+7;
    if (n2 > n) n2 = n;
   }

/* SEND TO FILE OR PRINTER */
printf("1) SEND TO FILE %s     2) SEND TO PRINTER     3)
EXIT",outname);
ch = getch();
printf("\n");

if (ch == '1')
 {

fprintf(outfile,"\n\n\n\311"); for (k=2; k<=67; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\273\n");
fprintf(outfile,"\272"); for (k=2; k<=67; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272                    VARIANCE-COVARIANCE MATRIX
                     \272\n");
fprintf(outfile,"\272"); for (k=2; k<=67; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");

n1 = 1;
if (n > 7) n2 = 7;
   else n2 = n;
while (n1 <= n)
   {

fprintf(outfile,"\307"); for (k=2; k<=67; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
for (k=1; k<=67; k++) fprintf(outfile," ");
fprintf(outfile,"\272\r");
fprintf(outfile,"\272        ");
for (i=n1; i<=n2; i++)  fprintf(outfile," X%-6.0f",beta[i][0]);
fprintf(outfile,"\n\307"); for (k=2; k<=67; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");

   for (i=1; i<=n; i++)
```

```
      { for (k=1; k<=67; k++) fprintf(outfile," ");
fprintf(outfile,"\272\r");
        fprintf(outfile,"\272  X%-5.0f",beta[i][0]);
        for (j=n1; j<=n2; j++)
            {fprintf(outfile,"%8.3f",scratch[i][j]);}
        fprintf(outfile,"\n");
      }
   fprintf(outfile,"\272"); for (k=2; k<=67; k++)
fprintf(outfile," "); fprintf(outfile,"\272\n");
   fprintf(outfile,"\310"); for (k=2; k<=67; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\274\n\n\n");
   n1 = n1+7;
    n2 = n2+7;
    if (n2 > n) n2 = n;
   }

 }
else if (ch == '2')
 {
   covarprt(scratch, beta, n);
 }


}

/* SEND COVARIANCE MATRIX TO PRINTER */

covarprt(scratch, beta, n)
float scratch[35][35], beta[35][2];
int n;
{

int i, j, k, n1, n2;
char ch;

fprintf(stdprn,"\n\n\n\r");
for (k=2; k<=67; k++) fprintf(stdprn,"\315");
fprintf(stdprn,"\r\n\n                          VARIANCE-COVARIANCE
MATRIX        \n\r");

n1 = 1;
if (n > 7) n2 = 7;
   else n2 = n;
while (n1 <= n)
   {

fprintf(stdprn,"\r\n");
for (k=2; k<=67; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\n\r          ");
   for (i=n1; i<=n2; i++)
```

```
            fprintf(stdprn," X%-6.0f",beta[i][0]);
fprintf(stdprn,"\r\n");
for (k=2; k<=67; k++) fprintf(stdprn,"\304");

   for (i=1; i<=n; i++)
     { fprintf(stdprn,"\r\n  X%-5.0f",beta[i][0]);
       for (j=n1; j<=n2; j++)
           {fprintf(stdprn,"%8.3f",scratch[i][j]);}
     }
   fprintf(stdprn,"\r\n\n");
   for (k=2; k<=67; k++) fprintf(stdprn,"\315");
   fprintf(stdprn,"\r\n\n\n");
   n1 = n1+7;
   n2 = n2+7;
   if (n2 > n) n2 = n;
   }
}


/* COVMAT CALCULATES THE VAR-COVARIANCE MATRIX */

covmat(msec,xpxinv,scratch,n)
float msec, xpxinv[35][35], scratch[35][35];
int n;
{
int i, j;

for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
        scratch[i][j] = msec * xpxinv[i][j];

}
```

REGOUT5

```
/* REGOUT5.C SUPPORTS LINEAR REGRESSION OUTPUT */

# include <stdio.h>
# include <math.h>
# include <dos.h>
# include <string.h>


/* XPXINVOUT OUTPUTS THE INV(X'X) MATRIX */

xpxinvout(xpxinv, beta, n, outname, outfile)
float xpxinv[35][35], beta[35][2];
int n;
char outname[24];
FILE *outfile;
{
int i, j, k, n1, n2;
char ch;


printf("\n\311"); for (k=2; k<=67; k++) printf("\315");
printf("\273\n");
printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");
printf("\272                        INV(X'X) MATRIX
            \272\n");
printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");

n1 = 1;
if (n > 8) n2 = 8;
   else n2 = n;
while (n1 <= n)
   {

printf("\307"); for (k=2; k<=67; k++) printf("\304");
printf("\266\n");
for (k=1; k<=67; k++) printf(" "); printf("\272\r");
printf("\272         ");
for (i=n1; i<=n2; i++)  printf(" X%-5.0f",beta[i][0]);
printf("\n\307"); for (k=2; k<=67; k++) printf("\304");
printf("\266\n");

   for (i=1; i<=n; i++)
     { for (k=1; k<=67; k++) printf(" "); printf("\272\r");
       printf("\272  X%-5.0f",beta[i][0]);
       for (j=n1; j<=n2; j++)
           {printf("%7.3f",xpxinv[i][j]);}
```

```
        printf("\n");
        if (i == 17 || i == 34)
           { printf("<RETURN>\r");  getch();  printf("
\r"); }
      }
   printf("\272"); for (k=2; k<=67; k++) printf(" ");
printf("\272\n");
   printf("\310"); for (k=2; k<=67; k++) printf("\315");
printf("\274\n\n");
   printf("<RETURN>\r");  getch();  printf("          \r");
   n1 = n1+8;
    n2 = n2+8;
    if (n2 > n) n2 = n;
   }

/*  SEND TO FILE OR PRINTER  */


printf("1) SEND TO FILE %s     2) SEND TO PRINTER     3)
EXIT",outname);
ch = getch();
printf("\n");

if (ch == '1')
 {

fprintf(outfile,"\n\311"); for (k=2; k<=67; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\273\n");
fprintf(outfile,"\272"); for (k=2; k<=67; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");
fprintf(outfile,"\272                        INV(X'X) MATRIX
                     \272\n");
fprintf(outfile,"\272"); for (k=2; k<=67; k++) fprintf(outfile,"
"); fprintf(outfile,"\272\n");

n1 = 1;
if (n > 8) n2 = 8;
   else n2 = n;
while (n1 <= n)
   {

fprintf(outfile,"\307"); for (k=2; k<=67; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
for (k=1; k<=67; k++) fprintf(outfile," ");
fprintf(outfile,"\272\r");
fprintf(outfile,"\272        ");
for (i=n1; i<=n2; i++)  fprintf(outfile," X%-5.0f",beta[i][0]);
fprintf(outfile,"\n\307"); for (k=2; k<=67; k++)
fprintf(outfile,"\304"); fprintf(outfile,"\266\n");
```

```
   for (i=1; i<=n; i++)
     { for (k=1; k<=67; k++) fprintf(outfile," ");
fprintf(outfile,"\272\r");
       fprintf(outfile,"\272  X%-5.0f",beta[i][0]);
       for (j=n1; j<=n2; j++)
           {fprintf(outfile,"%7.3f",xpxinv[i][j]);}
       fprintf(outfile,"\n");
     }
   fprintf(outfile,"\272"); for (k=2; k<=67; k++)
fprintf(outfile," "); fprintf(outfile,"\272\n");
   fprintf(outfile,"\310"); for (k=2; k<=67; k++)
fprintf(outfile,"\315"); fprintf(outfile,"\274\n\n\n");
   n1 = n1+8;
    n2 = n2+8;
    if (n2 > n) n2 = n;
    }

 }
else if (ch == '2')
 {

   xpxinvprt(xpxinv, beta, n);

 }

}

/* SEND XPXINV TO PRINTER */

xpxinvprt(xpxinv, beta, n)
float xpxinv[35][35], beta[35][2];
int n;
{
int i, j, k, n1, n2;
char ch;

fprintf(stdprn,"\n\n\n\r");
for (k=2; k<=67; k++) fprintf(stdprn,"\315");
fprintf(stdprn,"\r\n\n                        INV(X'X) MATRIX
    \n\r");

n1 = 1;
if (n > 8) n2 = 8;
   else n2 = n;
while (n1 <= n)
    {

fprintf(stdprn,"\r\n");
for (k=2; k<=67; k++) fprintf(stdprn,"\304");
fprintf(stdprn,"\n\r         ");
```

```
    for (i=n1; i<=n2; i++)
            fprintf(stdprn," X%-5.0f",beta[i][0]);
fprintf(stdprn,"\r\n");
for (k=2; k<=67; k++) fprintf(stdprn,"\304");

    for (i=1; i<=n; i++)
      { fprintf(stdprn,"\r\n  X%-5.0f",beta[i][0]);
        for (j=n1; j<=n2; j++)
            {fprintf(stdprn,"%7.3f",xpxinv[i][j]);}
      }
    fprintf(stdprn,"\r\n\n");
    for (k=2; k<=67; k++) fprintf(stdprn,"\315");
    fprintf(stdprn,"\r\n\n\n");
    n1 = n1+8;
     n2 = n2+8;
     if (n2 > n) n2 = n;
    }
}
```

Bibliography

1. Andriole, Stephen J. and others. "Storyboarding for C2 Systems Design: A Combat Support System Case Study," Proceedings of the Fifth Annual Workshop on Command & Control Decision Aiding, 27-29 October 1987.

2. Bauer, Maj Kenneth W. OPER 7.50: Response Surface Methodology Class Notes. School of Engineering, Air Force Institute of Technology, Fall 1989.

3. BBN Software Products Corporation. RS/DISCOVER User's Guide. Cambridge, MA, 1988.

4. Box, George E. P. and Norman R. Draper. Empirical Model-Building and Response Surfaces. New York: John Wiley & Sons, 1987.

5. Cohen, Marvin S. "When the Worst Case is Best: Mental Models, Uncertainty, and Decision Aids," American Psychological Association, December 1987.

6. Conte, S. D. and Carl de Boor, Elementary Numerical Analysis: An Algorithmic Approach, St. Louis: McGraw-Hill, 1980.

7. Davis, G.B. "Strategies for Information Requirements Determination," Information Analysis Selected Readings. Edited by Robert Galliers. Sydney: Addison-Wesley Publishing Company, 1987.

8. Dhar, Vasant. "On the Plausibility and Scope of Expert Systems in Management,"Journal of Management Information Systems, 4: 25-41 (Summer 1987).

9. Efron, Bradley. "Computer-Intensive Methods in Statistical Regression,"Siam Review, 30: 421-449 (September 1988).

10. Gale, William A. Artificial Intelligence and Statistics. Reading, MA: Addison Wesley Publishing Company, 1986.

11. Hahn, Gerald J. "More Intelligent Statistical Software and Statistical Expert Systems: Future Directions," The American Statistician, 39: 1-8 (February 1985).

12. Hippenmeyer, Capt William Frank. Adaptive Design of the Noncombatant Evacuation Operation Decision Support System for the U.S. Central Command. MS thesis. AFIT/GCS/ENG/88D-09. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB, OH May 1987.

13. Huber, George P. "Cognitive Style as a Basis for MIS and DSS Designs: Much Ado About Nothing?" Management Science Volume 24 No. 5: 567-579 (May 1983).

14. Keen, Peter G. W. "Adaptive Design for Decision Support Systems, "ACM/Database Volume 12 No. 192: 15-25 (Fall 1980).

15. Keen, Peter G. W. "Decision Support Systems: The Next Decade, "MIS Quarterly, 3: 253-265 (December 1988).

16. Kleijnen, Jack P. C. Statistical Tools for Simulation Practitioners, New York: Marcel Dekker, Inc., 1987.

17. Mellichamp, Joseph M. and Young H. Park. "A Statistical Expert System for Simulation Analysis," Simulation, 52: 134-139 (April 1989).

18. Myers, Raymond H. Response Surface Methodology. Virginia Polytechnic University, 1976.

19. Neter, John and others. Applied Linear Statistical Models (Second Edition). Homewood, Ill: Richard D. Irwin, Inc., 1985.

20. Press, William H. and others. Numerical Recipes, The Art of Scientific Computing. New York: Cambridge University Press, 1987.

21. Pritsker, A. Alan B. Introduction to Simulation and SLAM II. New York: John Wiley & Sons, 1986.

22. Robey, Daniel and William Taggart. "Human Information Processing in Information and Decision Support Systems," Decision Support Systems: a Data-based, Model-oriented, User-developed Discipline, edited by William C. House. New York: PBI, 1983.

23. Royston, J. P. "Expected Normal Order Statistics (Exact and Approximate," Royal Statistical Society: 161 (1982).

24. Schmidt, Stephen R. and Robert G. Launsby. Understanding Industrial Designed Experiments. Longmont, CO: CQG Ltd Printer, 1988.

25. Schildt, Herbert. C: Power User's Guide. Berkeley, CA: Osborne McGraw-Hill, 1988.

26. Shapiro, S. S. and R. S. Francia. "An Approximate Analysis of Variance Test for Normality," Journal of American Statistica Association, 67: 215-216 (March 1972).

27. Simon, Herbert A. "Decision Making and Problem Solving," Interfaces 17: 11-31 (September-October 1987).

28. Sparrow, Kalla J. An Interactive Computer Package for Use with Simulation Models which Performs Multidimensional Sensitivity Analysis by Employing the Techniques of Response Surface Methodology. MS Thesis. AFIT/GOR/OS/84D-12. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB, OH, December 1984 (151956).

29. Sprague, Ralph H. Jr. and Eric D. Carlson. Building Effective Decision Support Systems. Englewood Cliffs NJ: Prentice-Hall, 1982.

30. Valusek, Lt Col John R. "Adaptive Design of DSSs: A User Perspective," DSS-88 Transactions: Eighth International Conference on Decision Support Systems. Edited by E.S. Weber, Institute of Management Sciences, 1988.

31. Valusek, Lt Col John R. Class handout, "Concept Mapping," distributed in OPER 652, Decision Support Systems. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, August 1989.

32. Yielding, Capt Gary E. Automating Response Surface Methodology for the Arsenal Exchange Model. MS thesis. AFIT/GOR/ENS/86D-17. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB, OH, May 1987.

Vita

Captain Gregory J. Meidt [redacted]. He attended Lyle High school in Minnesota excelling as class valedictorian as well as a four sport letterman, graduating in 1979. He attended the U. S. Air Force Academy, Colorado, graduating with a Bachelor of Science in Operations Research in 1983.

From July 1983 to July 1988, Captain Meidt was assigned to the Antisatellite (ASAT) Program Office, Space Division, Los Angeles AFB, California. In this assignment, he held the duties of ASAT Test Data Analysis Manager, Chief of the ASAT Test Planning Branch, Chief of the ASAT Test Division, and Chief of the ASAT Budget and Estimation Branch. During this assignment he received several awards, including the Aerospace Primus Award for the successful first intercept of an orbiting satellite. In addition, he attended night school, receiving a Master of Business Administration from Pepperdine University in August 1986.

Captain Meidt entered the Air Force Institute of Technology School of Engineering in August 1988 and will be assigned as an instructor at the U. S. Air Force Academy Department of Management upon graduation.

He is married to [redacted]. They have a son, [redacted].

[redacted]

Vita

Captain David M. Leeper [redacted] He attended Lakeside High school in Atlanta, Georgia. He followed in his father's footsteps by attending the Citadel, the Military College of South Carolina, graduating with a Bachelor of Arts in Math in 1982.

From October 1982 to March 1986, he worked in the Defense Support Program (DSP) at Lowry Air Force Base Denver, Colorado. At this assignment, he held two different jobs. The first was as a satellite simulation specialist, and the second was as a member of select analyst group, solving program wide problems.

From March 1986 to August 1988, he worked as a member of the Air Force Operational Test Center (AFOTEC) test team, evaluating the Consolidated Space Operations Center (CSOC) in Colorado. He created the data management and analysis plan for the operational test of the CSOC. He also created a generic data base system to store and retrieve AFOTEC test data.

Captain Leeper entered the Air Force Institute of Technology School of Engineering in August 1988 and will be assigned to the Pentagon, XOX1 upon graduation.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/ENG/GOR90M-12	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A Micro-Computer Based Decision Support System for Response Surface Methodology

12. PERSONAL AUTHOR(S)
David M. Leeper, Capt, USAF and Gregory J. Meidt, Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM ____ TO ____	1990 March	348

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	04	
12	09	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Response Surface Methodology (RSM), Decision Support System (DSS), Experimental Design, Regression, Expert System

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Co-Advisors: John R. Valusek, Lt Col, USAF
Kenneth W. Bauer, Maj, USAF

Abstract: See Reverse

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Lt Col John R. Valusek, Maj Kenneth Bauer	22b. TELEPHONE (Include Area Code) 513-255-3362 22c. OFFICE SYMBOL AFIT/ENS

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE